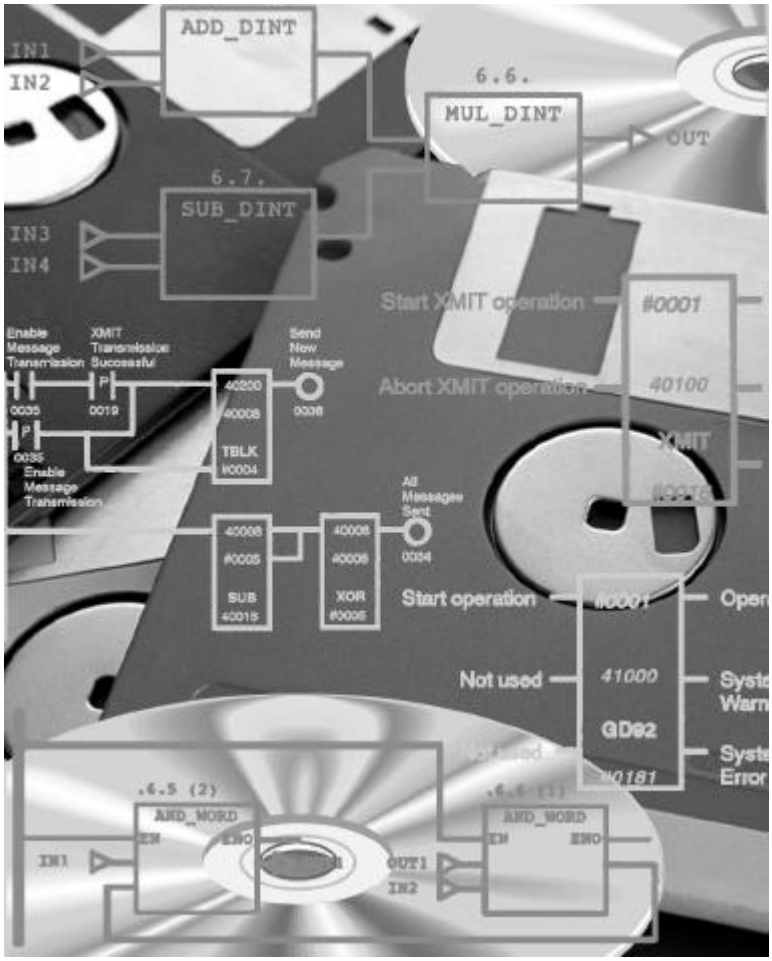
Modicon
XMIT Function Block
User Guide

840 USE 113 00 Version 3.0



GROUPE SCHNEIDER

■ Modicon ■ Square D ■ Telemecanique

Modicon
XMIT Function Block
User Guide

840 USE 113 00 Ver 3.0

August, 1998

Schneider Automation Inc.
One High Street
North Andover, MA 01845

Preface

The data and illustrations found in this book are not binding. We reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be construed as a commitment by Schneider Automation Inc.

Schneider Automation assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us by using the form on the last page of this publication.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of the Publisher, Schneider Automation Inc.

Caution: All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to assure compliance with documented system data, repairs to components should be performed only by the manufacturer.

MODSOFT® is a registered trademark of Schneider Automation Inc.

The following are trademarks of Schneider Automation Inc.:

Modbus	Modbus Plus
Modicon	984

DIGITAL®andDEC®areregisteredtrademarksofDigitalEquipment Corporation.

IBM® and IBM AT® are registered trademarks of International Business Machines Corporation.

Microsoft® and MS--DOS® are registered trademarks of Microsoft Corporation.

Procomm® is a registered trademarks of Datastorm Technologies Corporation.

© Copyright 1998, Schneider Automation Inc.
Printed in U.S.A.

Contents

Chapter 1
Introduction 1

1.1 XMIT Specific Functionality 2
1.1.1 Restrictions 2
1.1.2 Software and Hardware Required 3
1.1.3 Reference Documents 5
1.2 PLC Loadable Functions 6
1.3 XMIT Installation as a Loadable 7
1.3.1 DX Zoom Screens: Loading DXFDT.SYS 7
1.3.2 DX Zoom Screens: Loading XMIT.ZMM 7
1.3.3 Loading XMIT1968.HLP Help Screen File 8
1.3.4 Loading NSUP.EXE 8
1.3.5 Loading XMIT.EXE 9
1.4 XMIT Zoom Screens 13
1.4.1 DX Communication Screens (Eight) 13
1.4.2 DX Port Status Screen (Three) 18
1.4.3 DX Conversion Screen (Three) 19
1.5 Customer Service 21

Chapter 2
XMIT Communication Block 23

2.1 XMIT Communication Block 24
2.1.1 Characteristics 24
2.1.2 Representation 24
2.2 XMIT Communication Control Table 27
2.2.1 XMIT Revision Number (4x) Read Only 27
2.2.2 Communication Fault Status (4x + 1) Read Only 27
2.2.3 Available to User (4x + 2) 28
2.2.4 Data Rate (4x + 3) 29
2.2.5 Data Bits (4x + 4) 30
2.2.6 Parity Bits (4x + 5) 30
2.2.7 Stop Bits (4x + 6) 30
2.2.8 Available to User (4x + 7) 30
2.2.9 Command Word (4x + 8) 30

2.2.10 Message Pointer Word (4x + 9) 38
2.2.11 Message Length (4x + 10) 44
2.2.12 Response Time-Out (mS) (4x + 11) 44
2.2.13 Retry Limit (4x + 12) 44
2.2.14 Start of Transmission Delay (mS) (4x + 13) 45
2.2.15 End of Transmission Delay (mS) (4x + 14) 45
2.2.16 Current Retry (4x + 15) 45
2.3 XMIT Communication Functions 46

Chapter 3
XMIT Port Status Block 47

3.1 XMIT Port Status Block 48
3.1.1 Characteristics 48
3.1.2 Representation 48
3.2 XMIT Port Status Display Table 50
3.2.1 XMIT Revision Number (4x) Read Only 50
3.2.2 Port Status Fault Status (4x + 1) Read Only 50
3.2.3 Slave Login/Slave Port Active Status (4x + 2) Read Only . 50
3.2.4 Slave Transaction Counter (4x + 3) Read Only 51
3.2.5 Port State (4x + 4) Read Only 51
3.2.6 Input FIFO Status Bits (4x + 5) Read Only 51
3.2.7 Input FIFO Length (4x + 6) Read Only 52

Chapter 4
XMIT Conversion Block 53

4.1 XMIT Conversion Block 54
4.1.1 Characteristics 54
4.1.2 Representation 54
4.2 XMIT Conversion Control Table 57
4.2.1 XMIT Revision Number (4x) Read Only 57
4.2.2 Conversion Fault Status (4x + 1) Read Only 57
4.2.3 Available to User (4x + 2) 57
4.2.4 Data Conversion Control Bits (4x + 3) 58
4.2.5 Data Conversion Opcodes (4x + 4) 60
4.2.6 Source Register (4x + 5) 61
4.2.7 Destination Register (4x + 6) 61
4.2.8 ASCII String Character Count (4x + 7) 61

4.3 XMIT Conversion Opcode Examples 62
4.3.1 ASCII Input Related Conversion Examples 62
4.3.2 ASCII Output Related Conversion Examples 64
4.3.3 ASCII String Related Conversion Examples 65
4.4 Other Conversion Types 70
4.4.1 Binary to BCD Conversion 70
4.4.2 BCD to Binary Conversion 70

Chapter 5
Application Examples 71

5.1 Multiple Modbus Commands Transmission from PLC to PLC slave . . . 72
5.1.1 Hardware Configuration 72
5.1.2 Master PLC Setup 73
5.1.3 Ladder Logic 74
5.1.4 Conclusion 79
5.2 Fault Word Transmission to Slave PLC via Dialup Modems 80
5.2.1 Hardware Configuration 80
5.2.2 Modem Setup 80
5.2.3 Master PLC Setup 81
5.2.4 Ladder Logic 82
5.2.5 Conclusion 88

Appendix A
Technical References 89

A.1 Modbus Query/Response Parameter Limits 90
A.2 Cable Pinouts 93
A.2.1 9-pin (RS-232) to 25-pin (Modem) with no RTS/CTS 93
A.2.2 9-pin (RS-232) to 25-pin (Modem) with RTS/CTS 94
A.2.3 9-pin to 9-pin (Null Modem) 95
A.2.4 9-pin to 9-pin (Modem) 96
A.2.5 9-pin to 25-pin (Null Modem) 97
A.2.6 RJ45-(8x8) to 25-pin (Null Modem) 110XCA20401 98
A.2.7 RJ45-(8x8) to 9-pin (Null Modem)110XCA20301 99
A.2.8 RJ45-(8x8) to 25-pin (Modem)110XCA20401 100
A.2.9 RJ45-(8x8) to 25-pin (Modem)110XCA20401 101
A.2.10 RJ45-(8x8) to RJ45-(8x8) (Modem) 102
A.2.11 9-pin to RJ45-(8x8) (Modem)110XCA20301 103

A.2.12 9-pin to RJ45-(8x8) (Modem)110XCA20301 104
A.3 Cable Adapter Kits 105
A.4 XMIT Configuration using Hayes Compatible Dial-Up Modems (Only) 106
A.4.1 Initialization Message 106
A.4.2 Dial Message 108
A.4.3 Hangup Message 109

Index 111

Chapter 1
Introduction

- XMIT Specific Functionality
- PLC Loadable Functions
- XMIT Installation as a Loadable
- Customer Service

1.1 XMIT Specific Functionality

The XMIT (Transmit) function block sends Modbus messages from a ”master” PLC to multiple slave PLCs or sends ASCII character strings from the PLC’s Modbus slave port#1 or port#2 to ASCII printers and terminals. XMIT sends these messages over telephone dialup modems, radio modems, or simply direct connection.

XMIT comes with three modes: a communication mode, port status mode and a conversion mode. XMIT performs general ASCII input functions in the communication mode including simple ASCII and terminated ASCII. You may use an additional XMIT block for reporting port status information into registers while another XMIT block performs the ASCII communication function. You may import and export ASCII or binary data into your PLC and convert it into various binary data or ASCII to send to DCE devices based upon the needs of your application. Refer to Chapter Two for details about the XMIT communication block, Chapter Three for XMIT port status block and Chapter Four for XMIT conversion block.

The block has builtin diagnostics that checks to make sure no other XMIT blocks are active in the PLC. Within the XMIT block a control table allows you to control the communications link between the PLC and DCE (Data Communication Equipment) devices attached to Modbus port #1 or port#2 of the PLC. The XMIT block does NOT activate the port LED when it is transmitting data.

Expert: Remember, the Modbus protocol is a ”master/slave” protocol. Modbus is designed to have only one master polling multiple slaves. Therefore, when using the XMIT block in a network with multiple masters, contention resolution and collision avoidance is your responsibility and may easily be addressed through ladder logic programming.

1.1.1 Restrictions

This function block controls Modbus port #1 of the 140 CPU 113 02, 140 CPU 213 04, 140 CPU 113 03, 140 CPU 424 02, and PC E984 2xx PLCs. The XMIT does not control Modbus port #2 of these PLCs. In contrast, this function block controls Modbus port #2 of the 110 CPU 612 04. The XMIT does not control Modbus port #1 of this PLC. This function block does NOT operate on Modicon Micro models 110 CPU 512 xx, 110 CPU 612 00/03, and Compact models PC A984 1xx, PC 0984 1xx, PLCs. Limits exist for Modbus query/response parameters based upon the PLC model. Refer to Appendix A.

1.1.2 Software and Hardware Required

The XMIT function block is available in two forms. Either as a loadable or a builtin block. When using the Quantum, Compact or Micro PLCs it is a loadable. When using the Momentum PLCs it is a builtin.

1.1.2.1 Loadable Form

- Modsoft Version 2.5 or lower (Part Number SW MSxD 9SA). Refer to Section 1.3.1 for details.
- Modsoft Version 2.6 or higher (Part Number SW MSxD 9SA). Refer to Section 1.3.2 for details.
- XMIT Loadable Function Block (Part Number 309 COM 455 0x). This includes README.TXT, NSUP.EXE, XMIT1968.HLP, XMIT.EXE files plus two sub-directories; /MS_25 that contains the DXFDT.SYS file and /MS_26 that contains the XMIT.ZMM file.

☞ **Note:** These are not PC executable programs.

- Quantum PLCs: 140 CPU 113 02, 140 CPU 213 04, 140 CPU 113 03 (with 2.12 executive or higher), 140 CPU 424 02 (with 2.10 executive or higher), and 140 CPU 434 12, 140 CPU 534 14 (with 1.02 executive or higher).
- Compatible Compact PLCs: PC E984 241, PC E984 245, PC E984 251, and PC E984 255 (with 1.02 executive or higher).
- Compatible Micro PLCs: 110 CPU 612 04 (with 1.00 executive or higher).
- Communication media. For example, dialup type modem/lease line modems. Refer to the list of tested modems and printers in the *Modicon 309 COM 455 00 XMIT Loadable Read–Me–First* (GI–XMIT–RMF).

1.1.2.2 Builtin Form

- Modsoft Version 2.6 or higher (Part Number SW MSxD 9SA).
- Momentum PLCs: 171 CCS 700 00, 171 CCS 700 10, 171 CCS 760 00, 171 CCS 780 00, 171 CCC 760 10, and 171 CCC 780 10 (with 2.00 executive or higher).

- **V** Communication media. For example, dialup type modem/lease line modems. Refer to the list of tested modems and printers in the *Modicon 309 COM 455 00 XMIT Loadable Read−Me−First* (GI−XMIT−RMF).

1.1.3 Reference Documents

- *Modicon Modbus Protocol Reference Guide*
  (PI MBUS 300)

- *984 A120 Compact Programmable Controllers User Guide*
  (890 USE 108 00)

- *Modicon TSX Quantum Automation Series Hardware Reference Guide*
  (840 USE 100 00)

- *Modicon TSX Momentum M1 Processor Adapter and Option Adapter User Manual*
  (870 USE 101 00)

- *Modicon 512/612 Micro PLC Hardware User Manual*
  (890 USE 145 00)

- *Modicon Micro Controllers Ladder Logic Manual*
  (890 USE 146 00)

- *Modicon Modsoft Programmer Software User Guide*
  (890 USE 115 00)

- *Modicon Ladder Logic Block Library User Guide*
  (840 USE 101 00)

- *Modicon 309 COM 455 0x XMIT Loadable Read Me First*
  (GI XMIT RMF)

1.2 PLC Loadable Functions

The PLC has, within its configuration data tables, the ability to have Loadable Function Blocks added to the controller. These Loadable Functions are application specific programmable blocks that are loaded and allow you to set the opcode through Modsoft panel software and then ”configure” them into the larger control program.

The executable software code is programmed into the application in the format of a standard three node ladder logic instruction block. The basic logic subroutine structure of a loadable is shown in Figure 1.

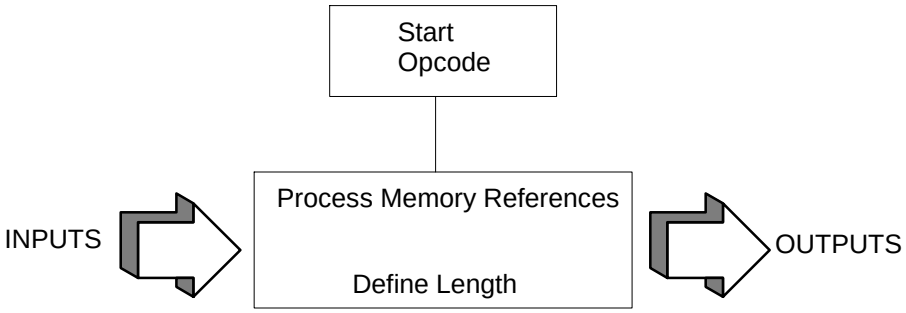


Figure 1 Loadable Code Logic Flow

These functions can be added to existing control logic in a field environment and offer a software solution to specific application problems.

1.3 XMIT Installation as a Loadable

This information assumes that you are familiar with Modsoft, have configured the PLC and are about to load XMIT.

When the loadable is transferred to the panel, Modsoft converts XMIT.EXE to a DX file named *XMIT1968.EXE*.

The next few figures are examples of the screens you see as you transfer the XMIT Loadable from the disk to the 984 controller. When you have concluded the transfer to the panel, the DX will be downloaded to the controller when you download the configuration.

Step 1 Insert the XMIT Loadable Function Block Disk (Part Number 309 COM 455 0x) into disk drive A:

Step 2 Go to the *Offline* (F2) selection on the Main Menu.

Step 3 Select either *Select Program* or *New Program* from the menu.

Step 4 Select *Configuration* (F5) from the menu.

☞ **Note:** Remember, the README.TXT, NSUP.EXE, XMIT1968.HLP, XMIT.EXE files plus two sub-directories; /MS_25 that contains the DXFDT.SYS file and /MS_26 that contains the XMIT.ZMM file are present on the disk. The NSUP and LSUP loadables are used to interface .EXE loadables with the PLC operating system.

1.3.1 DX Zoom Screens: Loading DXFDT.SYS

Only load DXFDT.SYS when using Modsoft 2.5 or lower for DX zoom screens. This file is in the /MS_25 sub-directory.

Step 1 Copy the DXFDT.SYS file to the Modsoft/runtime directory. The DXFDT.SYS file replaces the existing DXFDT.SYS file.

1.3.2 DX Zoom Screens: Loading XMIT.ZMM

Only load XMIT.ZMM when using Modsoft 2.6 or higher for DX zoom screens. This file is in the /MS_26 sub-directory.

Step 1 Copy the XMIT.ZMM file to the directory in which the program files reside. This file MUST be in the same directory as the program files for the program using XMIT, or the DX zoom screens will not be available.

1.3.3 Loading XMIT1968.HLP Help Screen File

Step 1 Copy the XMIT1968.HLP file to the directory in which the program files reside. This file MUST be in the same directory as the program files for the program using XMIT, or the help screen will not be available.

1.3.4 Loading NSUP.EXE

Step 1 Select *Loadable* (F7) then *Dir* (F3) then *Load* (F1). A prompt appears asking for the filename. Type *A:\ NSUP.EXE* and then hit return.

☞ **Note:** Either NSUP.EXE file or LSUP.EXE file must be loaded in order to run the XMIT Block on the PLC. Therefore, if you already have LSUP.EXE loaded, you need not load NSUP.EXE. Refer to Figure 2.

⚠ **Caution: The NSUP.EXE file *MUST* be loaded into the PLC *BEFORE* the XMIT.EXE file. If not the XMIT instruction will not operate correctly and all three outputs turn on.**

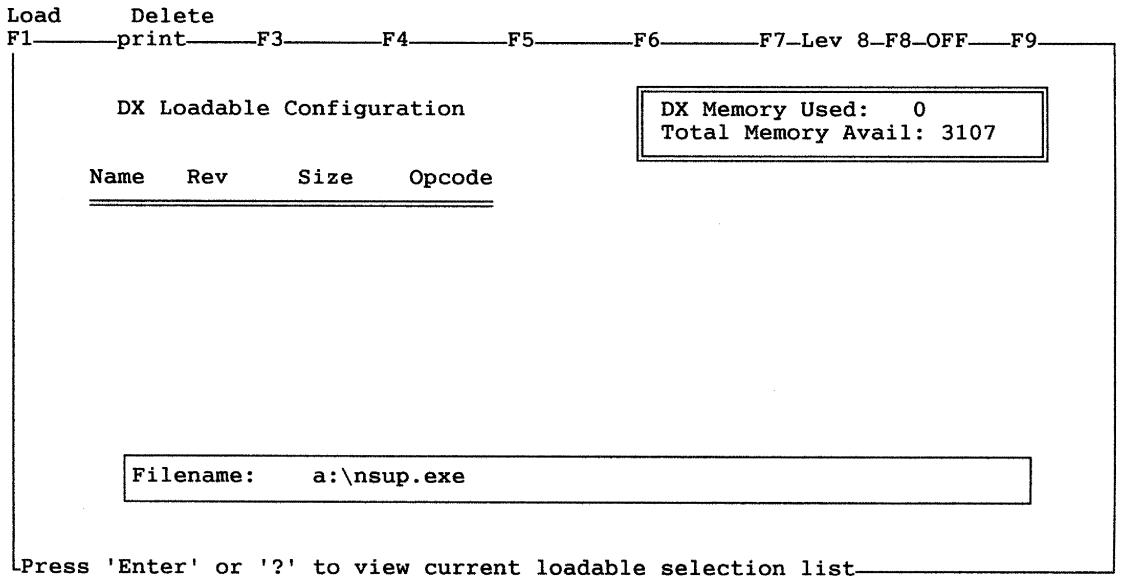


Figure 2 Loadable Screen (Sample Screen)

Step 2 A system message appears telling you that you can now access this loadable.

Step 3 Hit *Shift ?* to display all available loadables. The NSUP.EXE Loadable should now appear in this list.

Step 4 Place your cursor onto *NSUP.EXE* and press enter. This displays the revision, size, and opcode of the NSUP Loadable. Its Opcode is (ff Hex). Ensure that this opcode does not conflict with any other Opcodes that may be in use. If so, select a new opcode from the available list.

☞ **Note:** To obtain the latest revisions to your NSUP Loadable contact Customer Service.

1.3.5 Loading XMIT.EXE

⚠ **Caution: When the NSUP loadable is not installed or, installed after the XMIT loadable or, installed in a Quantum PLC with an older executive than specified in Section 2.1.1., all three outputs turn on regardless of the input states.**

⚠ **Caution: The NSUP.EXE file *MUST* be loaded into the PLC *BEFORE* the XMIT.EXE file. If not the XMIT instruction will not operate.**

Step 1 Select *Loadable* (F7) then *Dir* (F3) then *Load* (F1). A prompt appears asking for the filename. Type *A:\ XMIT.EXE* and then hit return. Refer to Figure 3.

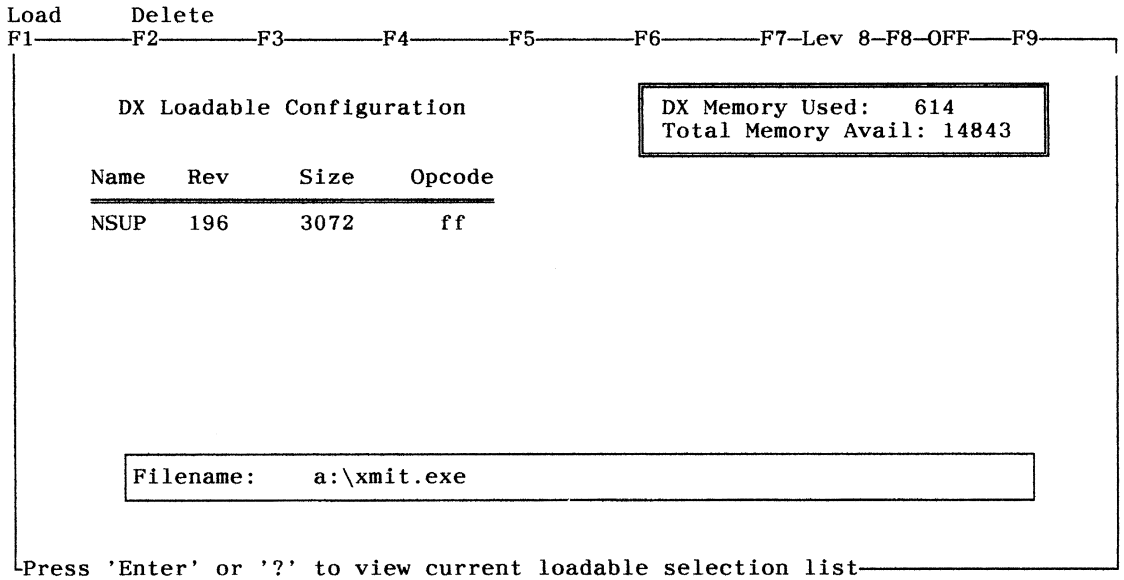


Figure 3 Loadable Screen (Sample Screen)

Step 2 System message appears telling you that you can now access this loadable. Move cursor below the name of the previous loadable to an open spot.

Step 3 Hit *Shift ?* to display all available loadables. The XMIT Loadable should now appears in this list. Refer to Figure 4.

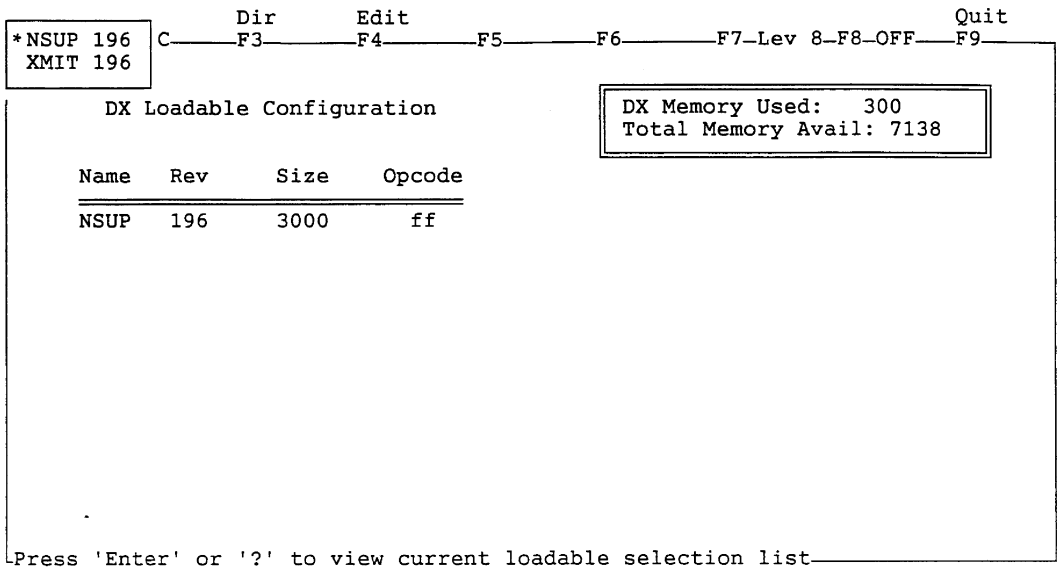


Figure 4 List of Available Loadables (Sample Screen)

Step 4 Place your cursor onto *XMIT.EXE* and press enter. This displays the revision, size and opcode of the XMIT Loadable. Its Opcode is (1e Hex). Ensure that this opcode does not conflict with any other opcodes that may be in use. The opcode shown on the screen may vary. Refer to Figure 5.

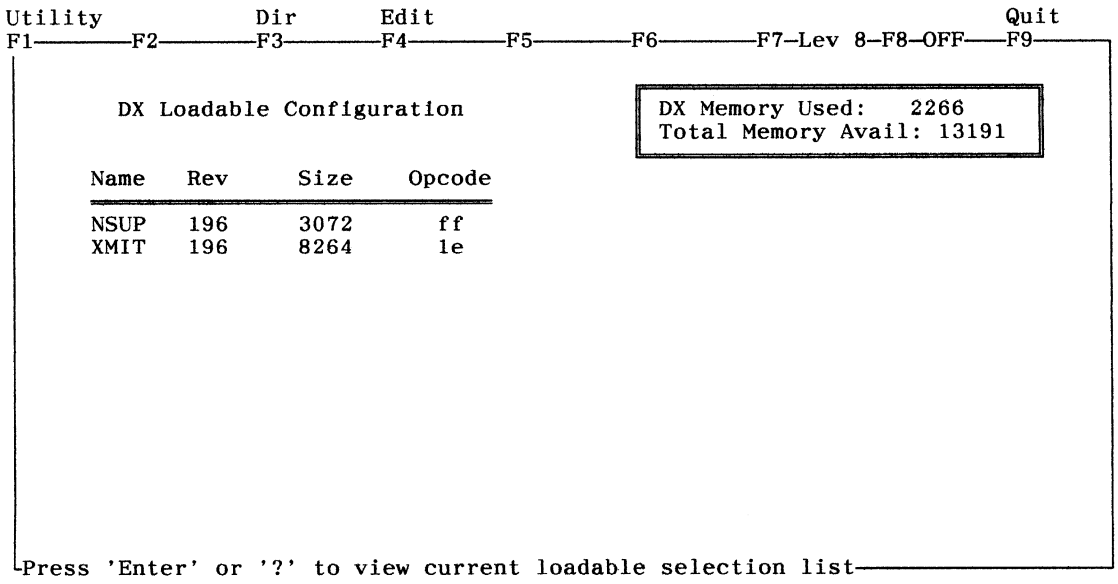


Figure 5 Installed Loadables (Sample Screen)

Step 5 Press *Escape* twice.

Step 6 The *Segment Status Display* appears. Select a segment, a network and press enter.

Step 7 Select *Element* (F3) from menu.

Step 8 Select *Loadable* (F5) from menu to access the XMIT Loadable.

Step 9 Pick your *XMIT* Loadable.

Step 10 Place your cursor onto the XMIT block and hit *ALT Z* to pull--up the XMIT zoom screens. At this point you may set your parameters based upon your application and the details of the XMIT Block found in Chapter Two. There are fourteen zoom screens for the XMIT block. Refer to Figure 6 through Figure 19.

☞ **Note:** Use *ALT H* to access the help screen for XMIT.

1.4 XMIT Zoom Screens

Ten zoom screens are used to set your parameters for both the loadable and builtin XMIT blocks and the details of the XMIT block are found in Chapter Two. There are three types of zoom screens: eight screens for XMIT communication mode, three for XMIT port status mode and three for XMIT conversion mode.

1.4.1 DX Communication Screens (Eight)

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4--- DX Zoom Editor --------F7-Lev 8-F8-OFF---F9------
|                            XMIT: Communication                    Page 1 / 8 |

4X   XMIT Revision Number                          40100  UINT  = 201          DEC
4X+1 Fault Status                                  40101  UINT  = 0            DEC
4X+2 Available To User:                            40102  UINT  = 0000         HEX
4X+3 Data Rate(50,75,110,134,150,300,600,1200,
     1800,2000,2400,3600,4800,7200,9600,19200)     40103  UINT  = 9600         DEC
4X+4 Data Bits (7,8):                              40104  UINT  = 8            DEC
4X+5 Parity (0=none; 1=odd; 2=even):               40105  UINT  = 2            DEC
4X+6 Stop Bits (1,2):                              40106  UINT  = 1            DEC
4X+7 Available To User:                            40107  UINT  = 0000         HEX
4X+8 Command Word:                                 40108  UINT  = 0100000000000000

B2 RTS/CTS Modm Ctrl:    ENABLED       | B3 RS485 MODE:          DISABLED
B5 Term'd Asc Input:     NO            | B6 Simple Asc Input:    NO
B7 ASCII Strng Msging:   DISABLED      | B8 Modbus Msging:       DISABLED
B9 Asc Recv FIFO:        DISABLED      | B10 Backspace:          DISABLED
B11 RTS/CTS Flow Ctrl:   DISABLED      | B12 Xon/Xoff Flow Ctrl:DISABLED
B13 Puls Dial Modm ATDP: NO            | B14 Hang-up Modm AT :   NO
B15 Tone Dial Modm ATDT: NO            | B16 Init Modem AT:      NO

                        Page up/down for prev/next screen
```

Figure 6 XMIT Communication Zoom Screen 1 of 8

```
Utility  PlcOps   Hex      Dec      Bin      Goto                          Quit
F1-------F2-------F3-------F4-- DX Zoom Editor --------F7-Lev 8-F8-ON-----F9-R1--
                            XMIT: Communication                  Page 2 / 8
PORT:  #0001
 4X+9 Message Pointer                          40109  UINT  = 150          DEC

 Function 01-06,15,16    Modbus function 08       Modbus functions 20,21
  4Y   Function code       4Y   Function code        4Y   Function code
  4Y+1 Quantity            4Y+1 Subfunction Code     4Y+1 Quantity
  4Y+2 Slave PLC address   4Y+2 Slave PLC address    4Y+2 Slave PLC address
  4Y+3 Slave data area     4Y+3 Diag. funct. data    4Y+3 Slave data area
  4Y+4 Master data area    4Y+4 Master data area     4Y+4 Master data area
                                                     4Y+5 File number

    Modbus Function Code Definitions
  01 Read Coil Status                 02 Read Input Status
  03 Read Holding Registers           04 Read Input Registers
  05 Force Single Coil                06 Preset Single Register
  08 Diagnostics
  15 Force Multiple Coils             16 Preset Multiple Registers
  20 Read general Reference (6X)      21 Write General Reference (6X)


                        Page up/down for prev/next screen
```

Figure 7 XMIT Communication Zoom Screen 2 of 8

```
Utility  PlcOps   Hex      Dec      Bin      Goto                          Quit
F1-------F2-------F3-------F4-- DX Zoom Editor --------F7-Lev 8-F8-ON-----F9-R1--
                  XMIT: Communication; Terminated Ascii Input    Page 3 / 8
PORT:  #0001
Ascii Message Pointer  40109  UINT  = 150          DEC
Ascii Message Length   40110  UINT  = 13           DEC
When Activated, Message pointer reg 4x+9 must have register offset to
the start of Ascii input definition table.  The table always has 5 words
and the message length register 4x+10 must contain the length of 5.

The internal format of the Ascii input definition table is as follows:

Word 0: High Byte = number of starting chars
        Low Byte = number of terminator chars
Word 1: High Byte = First starting char
        Low Byte = Second starting char
Word 2: High Byte = First terminator char
        Low Byte = Second terminator char
Word 3: Input storage destination reg offset, (e.g. 123=40123)
Word 4: Number of received chars written into input storage
        destination registers given by word 3.


                        Page up/down for prev/next screen
```

Figure 8 XMIT Communication Zoom Screen 3 of 8

```
Utility  PlcOps   Hex      Dec      Bin      Goto                          Quit
F1-------F2-------F3-------F4--- DX Zoom Editor --------F7-Lev 8-F8-ON-----F9-R1-
                              XMIT: Communication                  Page 4 / 8
PORT:  #0001
                          Diagnostics Code Definitions

  00 Return Query                              11 Return Bus Message Count
  01 Restart Comm Option                       12 Return Bus Comm. Error Cnt
  02 Return Diagnostic Register                13 Return Bus Exception Count
  03 Change ASCII Input Delimiter              16 Return Slave NAK Count
  04 Force Listen Only Mode                    17 Return Slave Busy Count
  10 Clear Counters                            18 Return Bus Char. Overrun Cnt



 4X+10 Message Length                          40110  UINT  = 13          DEC
 4X+11 Response Time-Out (ms)                  40111  UINT  = 30000       DEC
 4X+12 Retry Limit                             40112  UINT  = 3           DEC
 4X+13 Start of Transmission Delay (ms)        40113  UINT  = 0           DEC
 4X+14 End of Transmission Delay (ms)          40114  UINT  = 0           DEC
 4X+15 Current Retry                           40115  UINT  = 0           DEC

                       Page up/down for prev/next screen
```

Figure 9 XMIT Communication Zoom Screen 4 of 8

```
Utility  PlcOps   Hex      Dec      Bin      Goto                          Quit
F1-------F2-------F3-------F4--- DX Zoom Editor --------F7-Lev 8-F8-ON-----F9-R1-
                              XMIT: Communication                  Page 5 / 8
PORT:  #0001
  4X+1 Fault Status                            40101  UINT  = 0           DEC

 XMIT Fault Codes

   1 - Modbus exception - Illegal function
   2 - Modbus exception - Illegal data address
   3 - Modbus exception - Illegal data value
   4 - Modbus exception - Slave device failure
   5 - Modbus exception - Acknowledge
   6 - Modbus exception - Slave device busy
   7 - Modbus exception - Negative acknowledge
   8 - Modbus exception - Memory parity error
 100 - Slave PLC data area can not equal zero
 101 - Master PLC data area can not equal zero
 102 - Coil (0x) not configured
 103 - Holding register (4x) not configured
 104 - Data length can not equal zero
 105 - Pointer to message table can not equal zero

                       Page up/down for prev/next screen
```

Figure 10 XMIT Communication Zoom Screen 5 of 8

```
Utility  PlcOps   Hex      Dec      Bin      Goto                          Quit
F1-------F2-------F3-------F4-- DX Zoom Editor --------F7-Lev 8-F8-ON-----F9-R1--
                            XMIT: Communication                  Page 6 / 8
PORT:  #0001
  4X+1 Fault Status                              40101  UINT  = 0          DEC

 XMIT Fault Codes (Continued)

 106 - Pointer to message table outside the range of configured registers
 107 - Transmit message time-out
 108 - Undefined error
 109 - Modem returned ERROR
 110 - Modem returned NO CARRIER
 111 - Modem returned NO DIALTONE
 112 - Modem returned BUSY
 113 - Invalid LRC checksum from slave
 114 - Invalid CRC checksum from slave
 115 - Invalid Modbus function or subfunction
 116 - Modbus response message time-out
 117 - Modem reply time-out
 118 - XMIT could not gain access to PLC comm port


                       Page up/down for prev/next screen
```

Figure 11 XMIT Communication Zoom Screen 6 of 8

```
Utility  PlcOps   Hex      Dec      Bin      Goto                          Quit
F1-------F2-------F3-------F4-- DX Zoom Editor --------F7-Lev 8-F8-ON-----F9-R1--
                            XMIT: Communication                  Page 7 / 8
PORT:  #0001
  4X+1 Fault Status                              40101  UINT  = 0          DEC

 XMIT Fault Codes (Continued)

 119 - XMIT could not enable PLC port receiver
 120 - XMIT could not set PLC UART
 121 - User issued an abort command
 122 - Top node of XMIT is not equal to one
 123 - Bottom node of XMIT is not equal to sixteen
 124 - Undefined internal state
 125 - Broadcast mode not allowed with this Modbus function
 126 - DCE did not assert CTS
 127 - Illegal configuration(Data bits, Data rate, Parity or Stop bits)
 128 - Unexpected response received from Modbus slave
 129 - Invalid command word combination
 130   Command word changed while active
 131 - Invalid character count


                       Page up/down for prev/next screen
```

Figure 12 XMIT Communication Zoom Screen 7 of 8

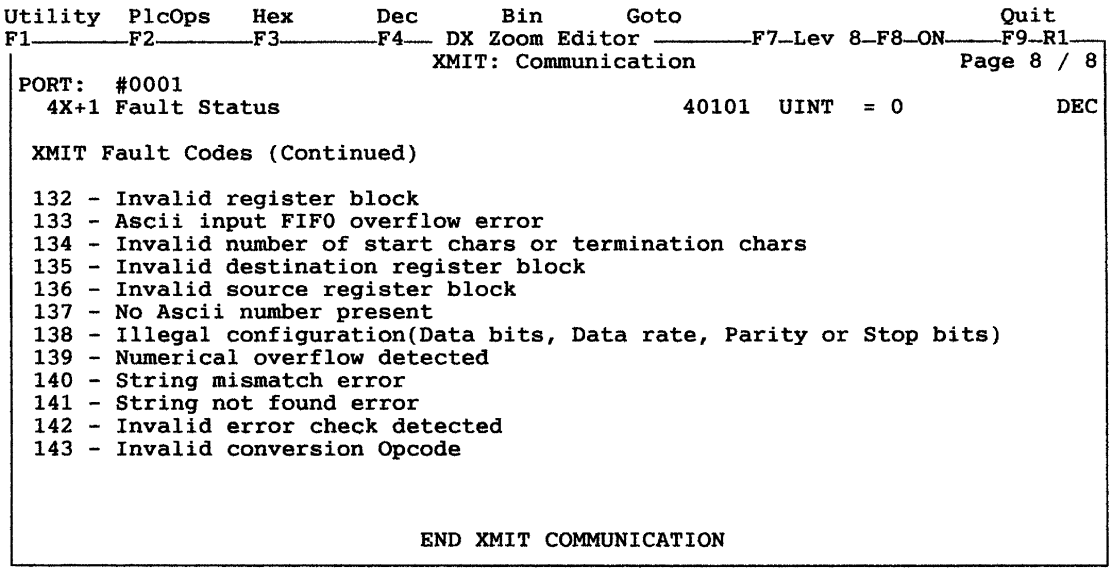

Utility PlcOps Hex Dec Bin Goto Quit
F1 F2 F3 F4 DX Zoom Editor F7 Lev 8 F8 ON F9 R1
XMIT: Communication Page 8 / 8
PORT: #0001
4X+1 Fault Status 40101 UINT = 0 DEC
XMIT Fault Codes (Continued)
132 - Invalid register block
133 - Ascii input FIFO overflow error
134 - Invalid number of start chars or termination chars
135 - Invalid destination register block
136 - Invalid source register block
137 - No Ascii number present
138 - Illegal configuration(Data bits, Data rate, Parity or Stop bits)
139 - Numerical overflow detected
140 - String mismatch error
141 - String not found error
142 - Invalid error check detected
143 - Invalid conversion Opcode
END XMIT COMMUNICATION

Figure 13 XMIT Communication Zoom Screen 8 of 8

1.4.2 DX Port Status Screen (Three)

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4-- DX Zoom Editor -------F7-Lev 8-F8-OFF--F9------
                              XMIT: GET STATUS                  PAGE 1 OF 3
PORT:  #0000
XMIT Revision Number          :40100  UINT  = 201         DEC

Fault Status                  :40101  UINT  = 0           DEC

Slave Logged-In (0=No;1=Yes) :40102  01:08 = 0           DEC
Slave Active(0=No;1=Yes)     :40102  09:16 = 0           DEC
Slave Transaction Counter    :40103  UINT  = 9600        DEC

Port State                    :40104  01:16 = 8           DEC

  Input FIFO Status Bits      :40105  UINT  = 0000000000000010
  Port Owned By(0=PLC;1=XMIT)                : 40105  04:04 = 0          DEC
  Ascii Output Blocked By Recvr(0=No;1=Yes) : 40105  08:08 = 0          DEC
  Ascii Inpt Has Blockd Send Dev(0=No;1=Yes): 40105  16:16 = 0          DEC
  Ascii Input FIFO Rcvd New Char(0=No;1=Yes): 40105  09:09 = 0          DEC
  Ascii Input FIFO Empty(0=No;1=Yes)         : 40105  10:10 = 0          DEC
  Ascii Input FIFO Enabled()=No;1=Yes)       : 40105  12:12 = 0          DEC
  Input FIFO Overflow Error(0=No;1=Yes)      : 40105  11:11 = 0          DEC
  Input FIFO Length                          : 40106  UINT  = 1          DEC
```

Figure 14 XMIT Port Status Zoom Screen 1 of 3

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4-- DX Zoom Editor -------F7-Lev 8-F8-OFF--F9------
                              XMIT: GET STATUS                  Page 2 / 3
PORT:  #0000
  4X+1 Fault Status                          40101  UINT  = 0          DEC

 XMIT Fault Codes (Continued)

 119 - XMIT Could Not Enable PLC Port Receiver
 120 - XMIT Could Not Set PLC UART
 121 - User Issued an Abort Command
 122 - Top node of XMIT is not equal to one
 123 - Bottom Node of XMIT is not equal to sixteen
 124 - Undefined Internal state
 125 - Broadcast Mode not Allowed with this Modbus Function
 126 - DCE did not Assert CTS
 127 - Illegal configuration(Data bits, Data rate, Parity or Stop bits)
 128 - Unexpected Response Received from Modbus Slave
 129 - Invalid Command Word Combination
 130 - Command Word Changed While Active
 131 - Invalid Character Count


                      Page up/down for prev/next screen
```

Figure 15 XMIT Port Status Zoom Screen 2 of 3

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4--- DX Zoom Editor ---------F7-Lev 8-F8-OFF---F9------
                                 XMIT: GET STATUS                  Page 3 / 3
PORT:  #0000
  4X+1 Fault Status                              40101  UINT  = 0          DEC

 XMIT Fault Codes (Continued)

 132 - Invalid Register Block
 133 - Ascii Input FIF0 Overflow Error
 134 - Invalid number of Start chars or Termination chars
 135 - Invalid Destination Register Block
 136 - Invalid Source Register Block
 137 - No Ascii Number Present
 138 - Illegal Configuration(Data bits, Data rate, Parity or Stop bits)
 139 - Numerical Overflow Detected
 140 - String Mismatch Error
 141 - String Not Found Error
 142 - Invalid Error Check Detected
 143 - Invalid Conversion Opcode


                        END XMIT GET STATUS COMMUNICATION
```

Figure 16 XMIT Port Status Zoom Screen 3 of 3

1.4.3 DX Conversion Screen (Three)

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4--- DX Zoom Editor ---------F7-Lev 8-F8-OFF---F9------
                                 XMIT: CONVERSIONS                 PAGE 1 OF 3

XMIT Revision Number:          40100  UINT  = 201          DEC
Fault Status:                  40101  UINT  = 0            DEC
Register Avail to User:        40102  UINT  = 0000         HEX
Data Conversion OpCode:                      COPY SRC BLK->DEST

Data Conversion Control Bits: 40103  UINT  = 0010010110000000
B2: CRC Seed:        0xFFFF          | B3: Err Chk Type:    LRC-8
B4: Error Check:     APPEND          |
B7: Conv. Case:      UPPR->LOWR      | B8: Case Sens.       NO
B9: Format Leading:  ZEROS           | B10: Output Format:  VARIABLE
B11: Conv. Type      SIGNED          | B12: Conv. Word:     16 BIT
B13: Auto Adv Src:   NO              | B14: Auto Adv Dest:  NO
B15: Begin Read Src: HI BYTE         | B16: Begin Save Dest:HI BYTE

SRC Register Offset :40105  UINT  = 2          DEC     Begin Read At:HI BYTE
                     (e.g. 100->40100)(Must be non-zero)
DEST Register Offset:40106  UINT  = 1          DEC     Begin Save At: HI BYTE
                     (e.g. 200->40200)(Must be non-zero)
Data Conversion Character Count:   40107  UINT  = 0          DEC
```

Figure 17 XMIT Conversion Zoom Screen 1 of 3

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4--- DX Zoom Editor --------F7-Lev 8-F8-OFF---F9------
                                 XMIT: CONVERSIONS                  Page 2 / 3
PORT:  #0000
  4X+1 Fault Status                              40101  UINT  = 0          DEC

 XMIT Fault Codes (Continued)

 119 - XMIT Could Not Enable PLC Port Receiver
 120 - XMIT Could Not Set PLC UART
 121 - User Issued an Abort Command
 122 - Top node of XMIT is not equal to one
 123 - Bottom Node of XMIT is not equal to sixteen
 124 - Undefined Internal state
 125 - Broadcast Mode not Allowed with this Modbus Function
 126 - DCE did not Assert CTS
 127 - Illegal configuration(Data bits, Data rate, Parity or Stop bits)
 128 - Unexpected Response Received from Modbus Slave
 129 - Invalid Command Word Combination
 130 - Command Word Changed While Active
 131 - Invalid Character Count


                       Page up/down for prev/next screen
```

Figure 18 XMIT Conversion Zoom Screen 2 of 3

```
Utility          Hex      Dec      Bin      Goto                          Quit
F1------MIKE------F3-------F4--- DX Zoom Editor --------F7-Lev 8-F8-OFF---F9------
                                 XMIT: CONVERSIONS                  Page 3 / 3
PORT:  #0000
  4X+1 Fault Status                              40101  UINT  = 0          DEC

 XMIT Fault Codes (Continued)

 132 - Invalid Register Block
 133 - Ascii Input FIFO Overflow Error
 134 - Invalid number of Start chars or Termination chars
 135 - Invalid Destination Register Block
 136 - Invalid Source Register Block
 137 - No Ascii Number Present
 138 - Illegal Configuration(Data bits, Data rate, Parity or Stop bits)
 139 - Numerical Overflow Detected
 140 - String Mismatch Error
 141 - String Not Found Error
 142 - Invalid Error Check Detected
 143 - Invalid Conversion Opcode


                       END XMIT CONVERSION COMMUNICATION
```

Figure 19 XMIT Conversion Zoom Screen 3 of 3

1.5 Customer Service

Schneider Automation telephone numbers are as follows:

- To call us from anywhere in North America except from within the state of Massachusetts: 1 (800) 468 5342
- To call us from within Massachusetts or from outside North America: 1 (978) 975 5001
- To fax us: 1 (978) 975 9301

Customer Service. When calling the Schneider Automation telephone number, ask for service from the list below.

When calling the 800 number, you will get a recording asking you to enter a one digit code for the type of service you want (listed below). However, this only works with a ”touch tone” phone. If using a dial phone, hang on and the operator will intercept after a short pause.

The service categories and *extra digit* code responses for push button phones are:

1 Technical support
2 Service order administration
3 Modfax
4 Training/course registration inquiries
5 General information other than above.

☞ **Note:** MODFAX: For available hardware data sheets, application notes, and software information. Recommended catalogue MC FAX DIR which is the master of all available catalogues (only twelve pages) lists all catalogues available on the MODFAX system.

☞ **Note:** BBS (Schneider Automation’s Customer Service Bulletin Board): For Modsoft updates, conversion utilities, hardware and software help, field service bulletins, Modbus and Modbus Plus help, software revision levels, FLASH EXEC updates for 984E controllers, loadable updates, and more. Parameters are up to 14.4k baud, no parity, 8 data, 1 stop, phone 1 (978) 975 9779.

☞ **Note:** Schneider Automation web page (www.modicon.com) provides user documentation, file updates, access to MODFAX, and other online services.

Chapter 2
XMIT Communication Block

- XMIT Communication Block
- XMIT Communication Control Table
- XMIT Communication Functions

2.1 XMIT Communication Block

The purpose of the XMIT communication block is to receive and transmit ASCII messages, and Modbus Master messages using your PLC ports.

2.1.1 Characteristics

Size

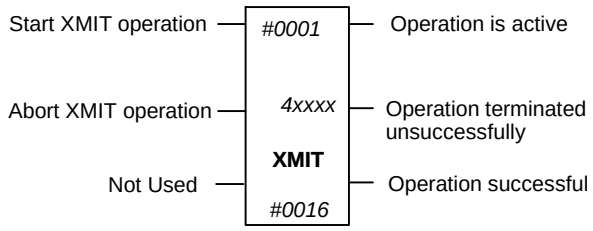
Three nodes high

PLC Compatibility

- Quantum PLCs: 140 CPU 113 02, 140 CPU 213 04, 140 CPU 113 03 (with 2.12 executive or higher), 140 CPU 424 02 (with 2.10 executive or higher), and 140 CPU 434 12, 140 CPU 534 14 (with 1.02 executive or higher).
- Compact PLCs: PC E984 241, PC E984 245, PC E984 251, and PC E984 255 (with 1.02 executive or higher).
- Micro PLCs: 110 CPU 612 04 (with 1.00 executive or higher).
- Momentum PLCs: 171 CCS 700 00, 171 CCS 700 10, 171 CCS 760 00, 171 CCS 780 00, 171 CCC 760 10, and 171 CCC 780 10 (with 2.00 executive or higher).

2.1.2 Representation

Block Structure



Inputs

XMIT has two possible control inputs. The input to the top node begins an XMIT operation and it should remain ON until the operation has completed successfully or an error has occurred. The input to the middle node aborts any active XMIT operation and forces the port to

slave mode. An abort code (121) is placed into the fault status register. The port remains closed as long as this input is ON.

☞ **Note:** To reset an XMIT fault and clear the fault register, the top input must go OFF for at least one PLC scan.

Outputs

XMIT may produce three possible outputs. The outputs from the top node goes ON while an XMIT operation is in progress. The output from the middle node goes ON when XMIT has detected an error or was issued an abort. The output from the bottom node goes ON when an XMIT operation has been successfully completed.

The following two cautions apply to loadables only.

⚠ **Caution: When the NSUP loadable is not installed or, installed after the XMIT loadable or, installed in a Quantum PLC with an older executive than specified in Section 2.1.1., all three outputs turn on regardless of the input states.**

⚠ **Caution: The NSUP.EXE file *MUST* be loaded into the PLC *BEFORE* the XMIT.EXE file. If not the XMIT instruction will not operate correctly and all three outputs turn on.**

Top Node Content

The top node must contain one of the following constants either (#0001) to select PLC port #1, or (#0002) to select PLC port #2. The loadable version does accept 4x registers in the top node, whereas the builtin does NOT.

Middle Node Content

The 4*x* register entered in the middle node is the first in a group of sixteen (16) contiguous holding registers that comprise the control block, as shown below:

XMIT Communication Control Table Description

Description	Register	Valid Entries
XMIT Revision Number	4x	Read Only
Fault Status	4x +1	Read Only
Available to User	4x +2	0 (*May be used as pointers for instructions like TBLK)*
Data Rate	4x +3	50, 75, 110, 134, 150, 300, 600, 1200, 2400, 9600, and 19200

Data Bits	4x +4	7,8
Parity	4x +5	0, 1, 2
Stop Bits	4x +6	0, 1, 2
Available to User	4x +7	*0 (May be used as pointers for instructions like TBLK)*
Command Word	4x +8	0000–0001–0000–0000 (256 Dec)
Pointer to Message Table	4x +9	Limited by the range of 4x registers configured
Length of Message	4x +10	0 ... 1024
Response Time–Out (mS)	4x +11	0 ... 65535
Retry Limit	4x +12	0 ... 65535
Start of Transmission Delay (mS)	4x +13	0 ... 65535
End of Transmission Delay (mS)	4x +14	0 ... 65535
Current Retry	4x +15	Read Only

Warning! DO NOT modify the address in the middle node of the XMIT block or delete it from the program while it is active. This locks up the port preventing communications.

Bottom Node Content

The bottom node must contain a constant equal to (#0016). This is the number of registers used by the XMIT instruction.

2.2 XMIT Communication Control Table

The following is a detail description of each of the (16) XMIT communication control table registers.

2.2.1 XMIT Revision Number (4x) Read Only

Displays the current revision number of XMIT block. This number is automatically loaded by the block and the block over writes any other number entered into this register.

2.2.2 Communication Fault Status (4x + 1) Read Only

This field displays a fault code generated by the XMIT block. A complete list is shown in the table below.

Fault Status (4x +1)

Fault Code	Fault Description
1	Modbus exception – Illegal function
2	Modbus exception – Illegal data address
3	Modbus exception – Illegal data value
4	Modbus exception – Slave device failure
5	Modbus exception – Acknowledge
6	Modbus exception – Slave device busy
7	Modbus exception –Negative acknowledge
8	Modbus exception –Memory parity error
9 ... 99	Reserved
100	Slave PLC data area cannot equal zero
101	Master PLC data area cannot equal zero
102	Coil (0x) not configured
103	Holding register (4x) not configured
104	Data length cannot equal zero
105	Pointer to message table cannot equal zero
106	Pointer to message table is outside the range of configured holding registers (4x)
107	Transmit message time–out (This error is generated when the UART cannot complete a transmission in 10 seconds or less. This error bypasses the retry counter and will activate the error output on the first error).
108	Undefined error
109	Modem returned ERROR

110	Modem returned NO CARRIER
111	Modem returned NO DIALTONE
112	Modem returned BUSY
113	Invalid LRC checksum from the slave PLC
114	Invalid CRC checksum from the slave PLC
115	Invalid Modbus function code
116	Modbus response message time−out
117	Modem reply time−out
118	XMIT could not gain access to PLC communications port #1 or port #2
119	XMIT could not enable PLC port receiver
120	XMIT could not set PLC UART
121	User issued an abort command
122	Top node of XMIT not equal to zero, one or two
123	Bottom node of XMIT is not equal to seven, eight or sixteen
124	Undefined internal state
125	Broadcast mode not allowed with this Modbus function code
126	DCE did not assert CTS
127	Illegal configuration (data rate, data bits, parity, or stop bits)
128	Unexpected response received from Modbus slave
129	Illegal command word setting
130	Command word changed while active
131	Invalid character count
132	Invalid register block
133	ASCII input FIFO overflow error
134	Invalid number of start characters or termination characters

2.2.3 Available to User (4x + 2)

The XMIT block does not use this register. However, it may be used in ladder logic as a pointer. An efficient way to use the XMIT block is to place a pointer value of a TBLK instruction into this register. For a example, refer to Chapter Five, Application Examples.

2.2.4 Data Rate (4x + 3)

XMIT supports the following data rates: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600 and 19200. To configure a data rate, enter its decimal number into this field. When an invalid data rate is entered, the block displays an illegal configuration error (error code 127) in the Fault Status (4x + 1) register.

2.2.5 Data Bits (4x + 4)

XMIT supports the following data bits: 7 and 8. To configure a data bit size, enter its decimal number into this register. Modbus messages may be sent in ASCII mode or RTU mode. ASCII mode requires 7 data bits, while RTU mode requires 8 data bits. When sending ASCII character message you may use either 7 or 8 data bits. When an invalid data bit is entered, the block displays an illegal configuration error (error code 127) in the Fault Status (4x + 1) register. For more details on Modbus message formats refer to *Modicon Modbus Protocol Reference Guide* (PI MBUS 300).

2.2.6 Parity Bits (4x + 5)

XMIT supports the following parity: none, odd and even. Enter a decimal of either: 0 = no parity, 1 = odd parity, or 2 = even parity. When an invalid parity is entered, the block displays an illegal configuration error (error code 127) in the Fault Status (4x + 1) register.

2.2.7 Stop Bits (4x + 6)

XMIT supports one or two stop bits. Enter a decimal of either: 1 = one stop bit, or 2 = two stop bits. When an invalid stop bit is entered, the block displays an illegal configuration error (error code 127) in the Fault Status (4x + 1) register.

2.2.8 Available to User (4x + 7)

The XMIT block does not use this register. However, it may be used in ladder logic as a pointer. An efficient way to use the XMIT block is to place a pointer value of a TBLK instruction into this register. For a example, refer to Chapter Five, Application Examples.

2.2.9 Command Word (4x + 8)

The XMIT interprets each bit of the command word as a function to perform. If bit 7 and 8 are on simultaneously or if any two or more of bits 13, 14, 15 or 16 are on simultaneously or if bit 7 is not on when bits 13, 14, 15, or 16 are on error 129 will be generated. Other restrictions apply. For more details refer to Section 2.3. The individual bit definitions are shown in the table below.

Command Word (4x +8) Bit Definitions

Bit 1 (msb)	Reserved
Bit 2 Enable RTS/ CTS modem control	Set to 1 when a DCE connected to the PLC requires hardware handshaking using RTS/CTS control. This bit may be used in conjunction with values contained in (4x + 13) and (4x + 14). Start of transmission delay (4x + 13) keeps RTS asserted for (X mS) before XMIT sends message out of PLC port #1. Likewise, end of transmission delay (4x + 14) keeps RTS asserted for (X mS) after XMIT has finished sending a message out of the PLC port #1. Once the end of transmission delay expires XMIT de−assert RTS.
Bit 3 Enable RS485 mode	Set to 1 when the selected port should operate in RS485 mode. Otherwise it defaults to 0, which is RS232 mode.
Bit 4	Reserved
Bit 5 Terminated ASCII input	Set to 1 to remove and discard all characters from FIFO until the starting string is matched, then these starting characters and subsequent characters are written into a contiguous 4x register destination block until the terminator sequence is matched. The terminator string is also written into the 4x register destination block. Refer to 2.2.9.1 for more details.
Bit 6 Simple ASCII input	Set to 1 to remove the ASCII characters from FIFO for writing into a contiguous 4x register block. The Message pointer (4x + 9) specifics the 4x register block. Refer to 2.2.9.2 for more details.
Bit 7 Enable ASCII string messaging	Set to 1 when you want to send ASCII messages out of the PLC. XMIT sends ASCII strings up to 1024 characters in length. You program each ASCII message into contiguous 4x registers of the PLC. Two characters allowed per register. Only use Bit 7 **OR** Bit 8, do not try to use both.
Bit 8 Enable Modbus messaging	Set to 1 when you want to send Modbus messages out of the PLC. Modbus messages may be in either RTU or ASCII formats. When data bits=8, XMIT uses Modbus RTU format. When data bits=7, XMIT uses Modbus ASCII format. Only use Bit 7 **OR** Bit 8, do not try to use both.
Bit 9 Enable ASCII receive FIFO	Set to 1 to allow the XMIT block to take control over the selected port (1 or 2) from the PLC. The block begins to receive ASCII characters into an empty 512 byte circular FIFO. Refer to 2.2.9.3 for more details.
Bit 10 Enable back space	Set to 1 to allow special handling of ASCII back space character (BS, 8Hex). When using either Simple ASCII Input (Bit 6) or Terminated ASCII Input (Bit 5) each back space character is removed from FIFO and may or may NOT be stored into a 4x register destination block. Refer to 2.2.9.4 for more details.
Bit 11 Enable RTS/ CTS flow control	Set to 1 to allow full duplex hardware flow control using the RTS and CTS handshaking signals for ASCII massaging. The RTS/CTS operates in both the input and output modes. Refer to 2.2.9.5 for more details.

Bit 12 Enable Xon/ Xoff flow control	Set to 1 to allow full duplex software flow control using the ASCII Xon character (DC1, 11 Hex) and the ASCII Xoff character (DC3, 13 Hex). The Xon/Xoff operates in both the input and output modes. Refer to 2.2.9.6 for more details.
Bit 13 Pulse dial modem	Set to 1 when using a Hayes compatible dial–up modem and you wish to pulse dial a telephone number. You program the phone number into contiguous 4x registers of the PLC. A pointer to these registers must be placed in control table register (4x + 9) and the length of the message in (4x + 10). Pulse dialed numbers are sent to the modem automatically preceded by ATDP and with carriage return <CR> and line feed <LF> appended. Since the dial message is an ASCII string, bit 7 must be ON prior to sending the number to be dialed.
Bit 14 hangup modem	Set to 1 when using a Hayes compatible dial–up modem and you want to hangup the modem. You must use ladder logic to turn this bit ON. Since the hangup message is an ASCII string, bit 7 must be ON prior to sending the message. Hang up messages are sent to the modem automatically preceded by +++AT and with carriage return <CR> and line feed <LF> appended. XMIT looks for a correct disconnect response from the modem before it turns ON the bottom output, noting a successful completion.
Bit 15 Tone dial modem	Set to 1 when using a Hayes compatible dial–up modem and you wish to tone dial a telephone number. You program the dial message into contiguous 4x registers of the PLC. A pointer to the dial message must be placed in control table register (4x + 9) and the length of the message in (4x + 10). Tone dial numbers are sent to the modem automatically preceded by ATDT and with carriage return <CR> and line feed <LF> appended. Since the dial message is an ASCII string, bit 7 must be ON prior to sending the number to be dialed.
Bit 16 Initialize modem	Set to 1 when using a Hayes compatible dial–up modem and you want to initialize the modem. You program the initialization message into contiguous 4x registers of the PLC. A pointer to the initialization message must be placed in control table register (4x + 9) and the length of the message in (4x + 10). All messages are sent to the modem automatically preceded by AT and with a carriage return <CR> and line feed <LF> appended. Since the initialization message is an ASCII string, bit 7 must be ON prior to sending the message.

2.2.9.1 Terminated ASCII Input Function

When (4x + 8, Bit 5) is actived for terminated ASCII Input messages, the message pointer (4x + 9) is the register offset to the first register of the ASCII input definition table. The terminated ASCII definition table is five registers long. Therefore, set the message length register (4x + 10) to five for successful XMIT operation. The terminated ASCII input definition table is shown in the table below. Enter your data into your ASCII input definition table using the reference section of Modsoft.

Terminated ASCII Input Definition Table

Word	High Byte	Low Byte
4x + 0	Number of starting characters (allowed content = 0, 1, 2)	Number of terminator characters (allowed content = 1, 2)
4x + 1	First starting character	Second starting character
4x + 2	First terminator character	Second terminator character
4x + 3	First 4x storage destination register	
4x + 4	Counter: counts the number of received characters written into the 4x storage destination registers	

During the process, (4x +4) of the ASCII input definition table holds a running count of characters written into the 4x destination register block. Once the terminated string is received the bottom output on the XMIT block goes ON and (4x +4) of the ASCII input definition table holds the total length of the received string including the starting and terminator strings. At this point the XMIT block stills owns the port and continues to save newly received characters into the ASCII receive FIFO, because the enable ASCII receive FIFO (4x + 8, Bit 9) is ON.

Using ladder logic, you can clear the simple ASCII input (4x + 8,Bit 6) before the next scan, while leaving the enable ASCII receive FIFO (4x + 8, Bit 9) ON. Thus, the current 4x register destination block is NOT over written by newer FIFO data, which is still collected in the FIFO. Using ladder logic, you can clear both bits for enable ASCII receive FIFO (4x + 8, Bit 9), and terminated ASCII input (4x + 8,Bit 5) to return port control back to the PLC.

When too many characters are written into the 4x register destination block with NO terminator detected, or the 4x register destination block is outside the allowed range for the configured PLC an error is reported in Fault Status (4x +1). The character limit is the smaller of 1024 or two times the sizes of the 4x register destination block.

We recommend you place the 4x register destination block for terminated ASCII input (4x + 8,Bit 5) past all other 4x registers used in the application to avoid being over written by ASCII input in case the terminator is absent. Also, you could allocate 512 registers for the 4x register destination block.

Terminated ASCII Example

Assume that XMIT is activated with the command word (4x + 8, Bit 9 and 5) set. Enable ASCII FIFO and terminated ASCII. The following ASCII string is received by the port: “AMScrlf$weight= 1245 GRAMScrlf$wei”. Refer to the ASCII Input Definition Table that shows the contents denoted by ( ) used in this example.

Terminated ASCII Input Definition Table Example (contents)

Word	High Byte	Low Byte
4x + 0	Number of starting characters (0x01)	Number of terminator characters (0x02)
4x + 1	First starting character ('$')	Second starting character (Not Used)
4x + 2	First terminator character ('cr')	Second terminator character ('lf")
4x + 3	First 4x storage destination register (101) = 400101	
4x + 4	Counter: counts the number of received characters written into the 4x storage destination registers (??????)	

The XMIT block becomes ACTIVE and then discards from the input FIFO the initial five characters, "AMScrlf", because they do not match the first starting character set to '$'. On the logic scan after the '$' is received, the XMIT block remains ACTIVE and it copies the '$' and subsequent characters into the 4x destination storage, updating (4x + 4) of the ASCII Input Definition Table with the count done so far, as the characters come in. After the final termination character is received the bottom output "Operation Successful" is activated and (4x + 4) of the ASCII Input Definition Table contains the total length equal to 0x0016. The 4x destination storage block, starting at 400101 contains: "$w", "ei", "gh", "t", "=", "12", "45", "G", "RA", "MS", "cflf". On the scan that the bottom output "Operation Successful" is activated, the already received characters from the next message, "$wei", that came in after the termination string, remains in the ASCII input FIFO. This gives the ladder logic the opportunity to turn off the Terminated ASCII input (4x + 8, Bit 5) before the next scan solve of XMIT for this port, keeping those characters in the FIFO until the PLC completes processing the current message, that might take several scans.

2.2.9.2 Simple ASCII Input Function

Two characters are stored in each 4x register. The first character transferred from FIFO is stored in the high byte of the first 4x register. The second character is transferred from FIFO is stored in the low byte of the first register. The third character is stored in the high byte of the second 4x register, and so on. The Message Length Register (4x + 10) contains the length of the message (1 ... 1024). Therefore, the Message Length Register (4x + 10) decreases as the characters are transferred from FIFO into the contiguous 4x register block. Once the entire message is transfered the Message Length Register (4x +10) restores its initial value and the XMITs Operation Successful output is activated.

☞ **Note:** When SImple ASCII Input (Bit 6) and ASCII Receive FIFO (Bit 9) remain set, new characters are constantously transferred from FIFO into the same 4x register block thus constantly over writing the previous characters stored into the 4x register block.

By clearing Simple ASCII Input (Bit 7) before the next scan in your ladder logic and setting ASCII Receive FIFO (Bit 9) you can still collect new characters and avoid this continuous overwriting of the 4x register block.By clearing both Simple ASCII Input (Bit 7) and ASCII Receive FIFO (Bit 9) using ladder logic you return control of the port (1 or 2) back to the PLC.

When the Message Length Register (4x +10) is 0 or more than 1024, or the 4x register block is outside the allowed range for the configured PLC an error is reported in Fault Status (4x +1).

2.2.9.3 Enable ASCII Receive FIFO

Setting this bit to 0 ends this function. When the FIFO receives 512 characters an internal overflow is set. When this occurs all subsequent characters are discarded, all ASCII input operations (simple and terminated) are ended, and the block returns an error until you toggle (Bit 9). When (Bit 9) is toggled, all data in the FIFO is discarded, both ASCII input control bits are ignored (Simple ASCII (Bit 6), Terminated ASCII (Bit 5)), and when no ASCII output controls are selected then the control of the port (1 or 2) is returned back to the PLC.

You need to set either Terminated ASCII (Bit 5) or Simple ASCII (Bit 6) to remove the ASCII characters from FIFO for processing. No more than one of the following three bits can be set simultaneously: Terminated ASCII (Bit 5), Simple ASCII (Bit 6), or ASCII Output (Bit 7).

Full duplex operation may be achieved by setting both ASCII Receive FIFO (BIT 9), and ASCII Output (Bit 7). This allows simple ASCII transmission out of the PLC while still receiving ASCII characters into FIFO. This is useful when working with dumb terminals. When ASCII Receive FIFO (Bit 9) is set none of the following ASCII output controls are allowed: Modbus Master Messaging (Bit 8), Pulse Dial Modem (Bit 13), Hangup Modem (Bit 14), Tone Dial Modem (Bit 15) and Initialize Modem (Bit 16).

2.2.9.4 Enable Back space

When a BS is detected it is NOT stored into the 4x register destination block, in fact it deletes the previous character and thus decreases the Terminated (Bit 5) Character Counter (4x + 4) of the ASCII Input Definition Table. In contrast, when a regular ASCII character is

detected it is stored into the 4x register destination block and the Terminated (Bit 5) Character Counter (4x + 4) of the ASCII Input Definition Table is increased.

☞ **Note:** Back spaces CANNOT delete characters from an empty 4x register destination block, thus the Terminated (Bit 5) Character Counter (4x + 4) of the ASCII Input Definition Table never goes below 0.

This special back space functionality along with internal echo enabled at the terminal are very useful for dealing with dumb terminals. A single Terminated ASCII Input XMIT block searching for "cr" is activated with ASCII Receive FIFO (Bit 9) and back space (Bit 10) set. No additional ladder logic is required while the you type and edit characters using the back space on the fly. When you type "cr" XMIT activates the bottom output "Operation Successful", and the corrected data is all lined up properly in the 4x register destination block.

2.2.9.5 Enable RTS/CTS Flow Control

The following pertains to the output mode. The XMIT state variable is set to BLOCKED when CTS is OFF and the receiving device indicates it cannot process additional characters. Likewise, The XMIT state variable is set to UNBLOCKED when CTS is ON and the receiving devices indicates it CAN process additional characters.

When transmission is UNBLOCKED and Simple ASCII Output (Bit 7) and RTS/CTS Flow Control (Bit 11) are set then the transmit output data is sent out in 16 byte packets. After all output packets are sent then the bottom output on the XMIT block goes ON "Operation Successful".

If during a transmission it suddenly becomes BLOCKED, only the remaining characters in the current output packet are sent, never exceeding 16 characters, and the XMIT block remains ACTIVE indefinitely. Only when the CTS in ON will the ASCII output resume sending all remaining output packets.

The following pertains to the input mode. Since RTS is an output signal, it can be used independently of the ASCII output transmit process, to BLOCK or UNBLOCK sending devices. When ASCII Receive FIFO (Bit 9) is set the RTS/CTS Flow Control works in the input mode. When ASCII Receive FIFO (Bit 9) is set and neither of the two ASCII inputs are set, Simple ASCII Input (Bit 6) or Terminated ASCII Input (Bit 5), the received characters will fill the FIFO in which they are inserted.

Mean time the RTS Flow Control (Bit 11) is ON allowing the sending device to proceed.

When the FIFO is more than three quarters full with characters the RTS Control Flow (Bit 11) is cleared to BLOCK the sending device. The RTS Control Flow (Bit 11) remains cleared until either Simple ASCII Input (Bit 6) or Terminated ASCII Input (Bit 5) have removed enough characters from the FIFO whereby reducing it to less than one quarter full of characters at which point the RTS Control Flow (Bit 11) is tuned ON.

☞ **Note:** The RTS/CTS Flow Control algorithm is different from RTS/CTS Modem Control. The former is related to full duplex receive buffer overflow. The latter deals with the transmit process gaining access to a shared transmission medium. Therefore, it is illegal to simultaneously request both of these RTS/CTS algorithms.

☞ **Note:** You CANNOT select any type of RTS/CTS Flow Control (Bit 11) handshaking when the port is in RS 485 Mode (Bit 3) because these signals do NOT exist in RS 485 mode.

2.2.9.6 Enable Xon/Xoff Flow Control

The following pertains to the output mode. The XMIT state variable is set to BLOCKED when Xoff character is received. Likewise the XMIT state variable is set to UNBLOCKED when Xon character is received. In neither case will Xon or Xoff be inserted into the FIFO.

When transmission is UNBLOCKED and Simple ASCII Output (Bit 7) and Xon/Xoff Flow Control (Bit 12) are set then the transmit output data is sent out in 16 byte packets. After all output packets are sent then the bottom output on the XMIT block goes ON “Operation Successful”.

If during a transmission it suddenly becomes BLOCKED, only the remaining characters in the current output packet are sent, never exceeding 16 characters, and the XMIT block remains ACTIVE indefinitely. Only when the next Xon character is received will the ASCII output resume sending all remaining output packets.

The following pertains to the input mode. Xon/Xoff may be used to BLOCK or UNBLOCK sending devices. When ASCII Receive FIFO (Bit 9) is set the Xon/Xoff Control Flow (Bit 12) works in the input mode. When ASCII Receive FIFO (Bit 9) is set and neither of the two ASCII inputs are set, Simple ASCII Input (Bit 6) or Terminated ASCII Input

(Bit 5), the received characters will fill the FIFO in which they are inserted.

When the FIFO is more than three quarter full with characters and additional characters are received the FIFO state variable is set to send XOFF character out the serial port after a delay of up to 16 character times BLOCKING the sender and clearing the FIFO state variable.

When all ASCII output functions are (Bits 8,13,14,15, and 16) OFF and the Xon/Xoff Flow Control (Bit 12) is ON the delay time defaults to 1 character time. In contrast, when all ASCII output functions are (Bits 8,13,14,15, and 16) ON and the Xon/Xoff Flow Control (Bit 12) is ON then the ASCII output is broken up into 16 byte packets. Thus, pending Xoff characters DO NOT have to wait more than 16 character times before BLOCKING the sender.

Once the sender has stopped transmission, the PLC eventually removes the characters form the FIFO using either Simple ASCII Input (Bit 6) or Terminated ASCII Input (Bit 7).

When FIFO becomes less than one quarter full with characters the FIFO state variable is set to send XON. Thus, sending a Xon character out the serial port to UNBLOCK the sender.

☞ **Note:** To prevent lockup due to a disconnected cable or other intermittent communication errors, when the sender is BLOCKED and did NOT receive the Xon character correctly we use the following algorithm. When FIFO becomes empty and no characters are subsequently received, then a steady stream of Xon characters are transmitted at the rate of once every 5 seconds.

☞ **Note:** The Xon/Xoff Flow Control (Bit 12) is different from the RTS/CTS Control Flow (Bit 11). The former uses transmitted Xon and Xoff characters to prevent receive buffer overflow in full duplex mode. The latter uses hardware shaking signals to accomplish the same goal. Therefore, it is illegal to simultaneously request both of these flow control algorithms because RTS/CTS Flow Control (Bit 11) Modem Control implies a half duplex network while Xon/Xoff Flow Control (Bit 12) implies a full duplex network.

2.2.10 Message Pointer Word (4x + 9)

You enter a pointer that points to the beginning of the message table. For ASCII character strings, the pointer is the register offset to the first register of the ASCII character string. Each register hold up to two

ASCII characters. Each ASCII string may be up to 1024 characters in length. For example, when you want to send 10 ASCII messages out of the PLC, you must program 10 ASCII characters strings into 4x registers of the PLC and then through ladder logic set the pointer to the start of each message after each successful operation of XMIT.

2.2.10.1 Modbus Function Codes (01 ... 06, 15 and 16)

For Modbus, messages, the pointer is the register offset to the first register of the Modbus definition table. The Modbus definition table for Modbus function code: 01, 02, 03, 04, 05, 06, 15 and 16 is five registers long and you must program it for successful XMIT operation. The Modbus definition table is shown in the table below.

Modbus Definition Table Function Codes (01 ... 06, 15 and 16)

Modbus function code (4y)	XMIT supports the following function codes: 01 = Read multiple coils (0x) 02 = Read multiple discrete inputs (1x) 03 = Read multiple holding registers (4x) 04= Read multiple input registers (3x) 05 = Write single coil (0x) 06 = Write single holding registers (4x) 15 = Write multiple coils (0x) 16 = Write multiple holding registers (4x)
Quantity (4y +1)	Enter the amount of data you want written to the slave PLC or read from the slave PLC. For example, enter 100 to read 100 holding registers from the slave PLC or enter 32 to write 32 coils to a slave PLC. There is a size limitation on quantity that is dependent on the PLC model. Refer to Appendix A for complete details on limits.
Slave PLC address (4y + 2)	Enter the slave Modbus PLC address. Typically the Modbus address range is 1 ... 247. To send a Modbus message to multiple PLCs, enter 0 for the slave PLC address. This is referred to as Broadcast Mode. Broadcast Mode only supports Modbus function codes that writes data from the master PLC to slave PLCs. Broadcast Mode does *NOT* support Modbus function codes that read data from slave PLCs.
Slave PLC data area (4y + 3)	For a read command, the slave PLC data area is the source of the data. For a write command, the slave PLC data area is the destination for the data. For example, when you want to read coils (00300 ... 00500) from a slave PLC, enter 300 in this field. When you want to write data from a master PLC and place it into register (40100) of a slave PLC, enter 100 in this field. Depending on the type of Modbus command (write or read), the source and destination data areas must be as defined in the Source and Destination Data Areas table below.
Master PLC data area (4y + 4)	For a read command, the master PLC data area is the destination for the data returned by the slave. For a write command, the master PLC data area is the source of the data. For example, when you want to write coils (00016 ... 00032) located in the master PLC to a slave PLC, enter 16 in the field. When you want to read input registers (30001 ... 30100) from a slave PLC and place the data into the master PLC data area (40100 ... 40199), enter 100 in this field. Depending on the type of Modbus command (write or read), the source and destination data areas must be as defined in the Source and Destination Data Areas table below.

Source and Destination Data Areas for Function Codes (01 ... 06, 15 and 16)

Function Code	Master PLC Data Area	Slave PLC Data Area
03 (Read multiple 4x)	4x (destination)	4x (source)
04 (Read multiple 3x)	4x (destination)	3x (source)
01 (Read multiple 0x)	0x (destination)	0x (source)
02 (Read multiple 1x)	0x (destination)	1x (source)
16 (Write multiple 4x)	4x (source)	4x (destination)
15 (Write multiple 0x)	0x (source)	0x (destination)
05 (Write single 0x)	0x (source)	0x (destination)
06 (Write single 4x)	4x (source)	4x (destination)

When you want to send 20 Modbus messages out of the PLC, you must program 20 Modbus definition tables and then through ladder logic increment the pointer to each definition table after each successful operation of XMIT, or you may program 20 separate XMIT blocks and then activate them one at a time through ladder logic.

2.2.10.2 Modbus Function Code (08)

For Modbus messages, the pointer is the register offset to the first register of the Modbus definition table. The Modbus definition table for Modbus function code: 08 is five registers long and you must program it for successful XMIT operation. The Modbus definition table is shown in the table below.

Modbus Definition Table Function Codes (08)

Modbus function code (4y)	XMIT supports the following function code: 08 = Diagnostics
Diagnostics (4y +1)	Enter the diagnostics subfunction code decimal value in this filed to perform the specific diagnostics function desired. The following diagnostic subfunctions are supported: Code Description 00 Return query data 01 Restart comm option 02 Return diagnostic register 03 Change ASCII input delimiter 04 Force listen only mode 05 ... 09 Reserved 10 Clear counters (& diagnostics registers in 384, 484) 11 Return bus messages count 12 Return bus comm error count 13 Return bus exception error count 14 ... 15 Not supported 16 Return slave NAK count 17 Return slave busy count 18 Return bus Char overrun count 19 ... 21 Not supported
Slave PLC address (4y + 2)	Enter the slave Modbus PLC address. Typically the Modbus address range is 1 ... 247. Function code 8 dose *NOT* support Broadcast Mode (Address 0)
Diagnostics function data field content (4y + 3)	You must enter the decimal value needed for the data area of the specific diagnostic subfunction. For subfunctions 02, 04, 10, 11, 12, 13, 16, 17 and 18 this value is automatically set to zero. For subfunctions 00, 01, and 03 you must enter the desired data field value. For more details, refer to *Modicon Modbus Protocol Reference Guide (PI–MBUS–300)*.
Master PLC data area (4y + 4)	For all subfunctions, the master PLC data area is the destination for the data returned by the slave. You must specify a 4x register that marks the beginning of the data area where the returned data is placed. For example, to place the data into the master PLC data area starting at (40100), enter 100 in this field. Subfunction 04 does NOT return a response. For more details, refer to *Modicon Modbus Protocol Reference Guide (PI–MBUS–300)*.

2.2.10.3 Modbus Function Codes (20, 21)

For Modbus, messages, the pointer is the register offset to the first register of the Modbus definition table. The Modbus definition table for Modbus function codes: 20 and 21 is six registers long and you must program it for successful XMIT operation. The Modbus definition table is shown in the table below.

Modbus Definition Table Function Codes (20, 21)

Modbus function code (4y)	XMIT supports the following function codes: 20 = Read general reference (6x) 21 = Write general reference (6x)
Quantity (4y +1)	Enter the amount of data you want written to the slave PLC or read from the slave PLC. For example, enter 100 to read 100 holding registers from the slave PLC or enter 32 to write 32 coils to a slave PLC. There is a size limitation on quantity that is dependent on the PLC model. Refer to Appendix A for complete details on limits.
Slave PLC address (4y + 2)	Enter the slave Modbus PLC address. Typically the Modbus address range is 1 ... 247. Function code 20 and 21 do *NOT* support Broadcast Mode (Address 0).
Slave PLC data area (4y + 3)	For a read command, the slave PLC data area is the source of the data. For a write command, the slave PLC data area is the destination for the data. For example, when you want to read registers (600300 ... 600399) from a slave PLC, enter 300 in this field. When you want to write data from a master PLC and place it into register (600100) of a slave PLC, enter 100 in this field. Depending on the type of Modbus command (write or read), the source and destination data areas must be as defined in the Source and Destination Data Areas table below. The lowest extended register is addressed as register ”zero” (600000). The lowest holding register is addressed as register ”one” (400001).
Master PLC data area (4y + 4)	For a read command, the master PLC data area is the destination for the data returned by the slave. For a write command, the master PLC data area is the source of the data. For example, when you want to write registers (40016 ... 40032) located in the master PLC to 6x registers in a slave PLC, enter 16 in the filed. When you want to read 6x registers (600001 ... 600100) from a slave PLC and place the data into the master PLC data area (40100 ... 40199), enter 100 in this field. Depending on the type of Modbus command (write or read), the source and destination data areas must be as defined in the Source and Destination Data Areas table below. The lowest extended register is addressed as register ”zero” (600000). The lowest holding register is addressed as register ”one” (400001).
File number (4y + 5)	Enter the file number for the 6x registers to be written to or read from. (1 ... 10) depending on the size of the extended register data area. 6**0**0001 is 60001 file **1** and 6**9**0001 is 60001 file **10** as viewed by the Reference Data Editor in Modsoft.

Source and Destination Data Areas for Function Codes (20, 21)

Function Code	Master PLC Data Area	Slave PLC Data Area
20 (Read general reference 6x)	4x (destination)	6x (source)
21 (Write general reference 6x)	4x (source)	6x (destination)

When you want to send 20 Modbus messages out of the PLC, you must program 20 Modbus definition tables and then through ladder logic increment the pointer to each definition table after each successful operation of XMIT, or you may program 20 separate XMIT blocks and then activate them one at a time through ladder logic.

2.2.11 Message Length (4x + 10)

You enter the length of the current message. When XMIT is sending Modbus messages for function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16, the length of the message is automatically set to five. When XMIT is receving Terminated ASCII input the length of the message must be set to five or an error results. When XMIT is sending Modbus messages for function codes twenty and twenty--one, the length of the message is automatically set to six. When XMIT is sending ASCII messages, the length may be 1 ... 1024 ASCII characters per message.

2.2.12 Response Time–Out (mS) (4x + 11)

You enter the time value in milliseconds (mS) to determine how long XMIT waits for a valid response message from a slave device (PLC, modem, etc.). In addition, the time applies to ASCII transmissions and flow control operations. When the response message is not completely formed within this specified time, XMIT issues a fault. The valid range is 0 ... 65535 mS. The timeout is initiated after the last character in the message is sent.

2.2.13 Retry Limit (4x + 12)

You enter the quantity of retries to determine how many times XMIT sends a message to get a valid response from a slave device (PLC, modem, etc.). When the response message is not completely formed within this specified time, XMIT issues a fault and a fault code. The valid range is 0 ... 65535 # of retries. This field is used in conjunction with response time--out (4x + 11).

2.2.14 Start of Transmission Delay (mS) (4x + 13)

You enter the time value in milliseconds (mS) when RTS/CTS control is enabled, to determine how long XMIT waits after CTS is received before it transmits a message out of the PLC port #1. Also, you may use this register even when RTS/CTS is *NOT* in control. In this situation, the entered time value determines how long XMIT waits before it sends a message out of the PLC port #1. You may use this as a pre message delay timer. The valid range is 0 ... 65535 mS.

2.2.15 End of Transmission Delay (mS) (4x + 14)

You enter the time value in milliseconds (mS) when RTS/CTS control is enabled, to determine how long XMIT keeps RTS asserted once the message is sent out of the PLC port #1. After the time expires, XMIT deassert RTS. Also, you may use this register even when RTS/CTS is *NOT* in control. In this situation, the entered time value determines how long XMIT waits after it sends a message out of the PLC port #1. You may use this as a post message delay timer. The valid range is 0 ... 65535 mS.

2.2.16 Current Retry (4x + 15)

The value displayed here indicates the current number of retry attempts made by the XMIT block. This register is read only.

2.3 XMIT Communication Functions

The XMIT communication block performs six functions shown below. For each function certain bits of the command word (4x + 8) must be set.

(4x + 8) Command Word Functions in Relation to Bits

(4x + 8) Command Word Function	Command word bits that must be set to 1	Bits that MUST be set to = 0
Terminated ASCII input (Bit 5=1) *	2,3,9,10,11,12	6,7,8,13,14,15,16
Simple ASCII input (Bit 6=1) *	2,3,9,10,11,12	5,7,8,13,14,15,16
Simple ASCII output (Bit 7=1)	2,3,9,10,11,12	5,6,8,13,14,15,16
Modem output (Bit 7=1)	2,3,13,14,15,16	5,6,8,9,10,11,12 (plus one, but ONLY one, of the following bits is set to 1: 13,14,15 or 16, while the other three bits must be set to 0)
Modbus master messaging output (Bit 8=1)	2,3	5,6,7,9,10,11,12,13,14,15,16
Enable ASCII receive input FIFO ONLY (Bit 9=1)	2,3,10,11,12	5,6,7,8,13,14,15,16
* When using either of these functions you MUST set Enable ASCII receive FIFO ((4x + 8,Bit 9) to 1.		

Chapter 3
XMIT Port Status Block

- XMIT Port Status Block
- XMIT Port Status Display Table

3.1 XMIT Port Status Block

The XMIT port status block shows the current port status, Modbus slave activity, ASCII input FIFO and flow control information that may be used in ladder logic for some applications. The XMIT port status block is totally passive. It does not take, release, or control the PLC port.

3.1.1 Characteristics

Size

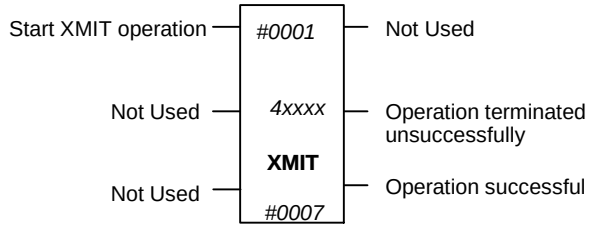
Three nodes high

PLC Compatibility

- Quantum PLCs: 140 CPU 113 02, 140 CPU 213 04, 140 CPU 113 03 (with 2.12 executive or higher), 140 CPU 424 02 (with 2.10 executive or higher), and 140 CPU 434 12, 140 CPU 534 14 (with 1.02 executive or higher).
- Compact PLCs: PC E984 241, PC E984 245, PC E984 251, and PC E984 255 (with 1.02 executive or higher).
- Micro PLCs: 110 CPU 612 04 (with 1.00 executive or higher).
- Momentum PLCs: 171 CCS 700 00, 171 CCS 700 10, 171 CCS 760 00, 171 CCS 780 00, 171 CCC 760 10, and 171 CCC 780 10 (with 2.00 executive or higher).

3.1.2 Representation

Block Structure



Inputs

XMIT has one possible control input. The input to the top node begins an XMIT operation and it should remain ON until the operation has

completed successfully or an error has occurred. The input to the middle node is not used on the XMIT port status block.

Note: To reset an XMIT fault and clear the fault register, the top input must go OFF for at least one PLC scan.

Outputs

XMIT may produce two possible outputs. The output from the top node is not used on the XMIT port status block. The output from the middle node goes ON when XMIT has detected an error or was issued an abort. The output from the bottom node goes ON when an XMIT operation has been successfully completed.

Top Node Content

The top node must contain one of the following constants either (#0001) to select PLC port #1, or (#0002) to select PLC port #2. The loadable version does accept 4x registers in the top node, whereas the builtin does NOT.

Middle Node Content

The 4*x* register entered in the middle node is the first in a group of seven (7) contiguous holding registers that comprise the port status display block, as shown below:

XMIT Port Status Control Table Description

Description	Register	No Valid Entries
XMIT Revision Number	4x	Read Only
Fault Status	4x +1	Read Only
Slave login status/ Slave port active status	4x +2	Read Only
Slave transaction counter	4x +3	Read Only
Port state	4x +4	Read Only
Input FIFO status bits	4x +5	Read Only
Input FIFO length	4x +6	Read Only

Warning! DO NOT modify the address in the middle node of the XMIT block or delete it from the program while it is active. This locks up the port preventing communications.

Bottom Node Content

The bottom node must contain a constant equal to (#0007). This is the number of registers used by the XMIT port status instruction.

3.2 XMIT Port Status Display Table

The following is a detail description of each of the (7) XMIT display table registers.

3.2.1 XMIT Revision Number (4x) Read Only

Displays the current revision number of XMIT block. This number is automatically loaded by the block and the block over writes any other number entered into this register.

3.2.2 Port Status Fault Status (4x + 1) Read Only

This field displays a fault code generated by the XMIT port status block. A complete list is shown in the table below.

Fault Status (4x +1)

Fault Code	Fault Description
118	XMIT could not gain access to PLC communications port #1 or port #2
122	Top node of XMIT not equal to zero, one or two
123	Bottom node of XMIT is not equal to seven, eight or sixteen

3.2.3 Slave Login Status/Slave Port Active Status (4x + 2) Read Only

This register displays status on two items generated by the XMIT port status block. Ladder logic may be able to use this information to reduce or avoid collisions on a multi master Modbus network.

(4x +2 high byte) Slave Login Status	(4x +2 low byte) Port Active Status
Yes= When a programming device is currently logged ON to this PLC slave port	Yes= When observed port is owned by the PLC and currently receiving a Modbus command OR transmitting a Modbus response
No= When a programming device is currently NOT logged ON to this PLC slave port (NOTE: A Modbus master can send commands but, not be logged ON)	No= When observed port is NOT owned by the PLC and currently receiving Modbus command OR transmitting a Modbus response

3.2.4 Slave Transaction Counter (4x + 3) Read Only

This register displays the number of slave transactions generated by the XMIT port status block. The counter increases every time the PLC Modbus slave port receives another command from the Modbus master. Ladder logic may be able to use this information to reduce or avoid collisions on a multi master Modbus network.

3.2.5 Port State (4x + 4) Read Only

This register displays ownership of the port and its state. It is generated by the XMIT port status block.

(4X + 4) Port State Options

Owns Port	Active State	Value
PLC	PLC Modbus slave	0
XMIT	Tone dial modem	1
XMIT	Hangup modem	2
XMIT	Modbus messaging	3
XMIT	Simple ASCII output	4
XMIT	Pulse dial modem	5
XMIT	Initialize modem	6
XMIT	Simple ASCII input	7
XMIT	Terminated ASCII input	8
XMIT	ASCII input FIFO is ON but, NO XMIT function is active	9

3.2.6 Input FIFO Status Bits (4x + 5) Read Only

The register displays the status of seven items related to the input FIFO. It is generated by the XMIT port status block.

(4x + 5) Input FIFO Status Bits and their Definitions

Bit #	Definition	Yes/1=	No/0=
1 ... 3	Not Used		
4	Port owned by...	XMIT	PLC
5 ... 7	Not Used		
8	ASCII output transmission...	Blocked by receiving device	Unblocked by receiving device
9	ASCII input received...	New character	NO new character

10	ASCII input FIFO is ...	Empty	Not empty
11	ASCII input FIFO is...	Overflowing (error)	Not overflowing (error)
12	ASCII input FIFO is...	On	Off
13 ... 15	Not Used		
16	ASCII input reception...	XMIT Blocked sending device	XMIT Unblocked sending device

3.2.7 Input FIFO Length (4x + 6) Read Only

This register displays the current number of characters present in the ASCII input FIFO. The register may contain other values based on the state of the input FIFO and if the length is empty or overflowing. It is generated by the XMIT port status block.

(4x + 6) Other Possible Values

When Input FIFO	Then Length
= OFF	= 0
= ON & Empty	= 0
= ON & Overflowing	= 512

Chapter 4
XMIT Conversion Block

- XMIT Conversion Block
- XMIT Conversion Control Table
- XMIT Conversion Examples
- Other Conversion Types

4.1 XMIT Conversion Block

The purpose of the XMIT conversion block is to take data and convert it into other usable forms based upon your application needs. The convert block performs eleven different functions. Some functions include ASCII to binary, integer to ASCII, byte swapping, searching ASCII strings, and others. The block allows internal conversions using 4x source blocks to 4x destination blocks.

4.1.1 Characteristics

Size

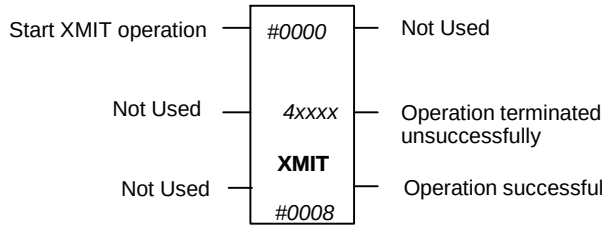
Three nodes high

PLC Compatibility

- Quantum PLCs: 140 CPU 113 02, 140 CPU 213 04, 140 CPU 113 03 (with 2.12 executive or higher), 140 CPU 424 02 (with 2.10 executive or higher), and 140 CPU 434 12, 140 CPU 534 14 (with 1.01 executive or higher).
- Compact PLCs: PC E984 241, PC E984 245, PC E984 251, and PC E984 255 (with 1.02 executive or higher).
- Micro PLCs: 110 CPU 612 04 (with 1.00 executive or higher).
- Momentum PLCs: 171 CCS 700 00, 171 CCS 700 10, 171 CCS 760 00, 171 CCS 780 00, 171 CCC 760 10, and 171 CCC 780 10 (with 2.00 executive or higher).

4.1.2 Representation

Block Structure



Inputs

XMIT has one possible control input. The input to the top node begins an XMIT operation and it should remain ON until the operation has completed successfully or an error has occurred. The input to the middle node is not used for the XMIT conversion block

☞ **Note:** To reset an XMIT fault and clear the fault register, the top input must go OFF for at least one PLC scan.

Outputs

XMIT may produce two possible outputs. The output from the top node is not used on the XMIT conversion block. The output from the middle node goes ON when XMIT has detected an error or was issued an abort. The output from the bottom node goes ON when an XMIT operation has been successfully completed.

Top Node Content

The top node must contain a constant (#0000) since conversions do not deal with the PLCs port. The loadable version does accept 4x registers in the top node, whereas the builtin does NOT.

Middle Node Content

The 4*x* register entered in the middle node is the first in a group of eight (8) contiguous holding registers that comprise the control block, as shown below:

XMIT Conversion Control Table Description

Description	Register	Valid Entries
XMIT Revision Number	4x	Read Only
Fault Status	4x +1	Read Only
Available to User	4x +2	0 (*May be used as pointers for instructions like TBLK)*
Data Conversion Control Bits	4x +3	Refer to the bit definition table for 4x + 3.
Data Conversion Opcode	4x +4	Refer to the definition table for 4x + 4.
Source Register	4x +5	4x register (begin read at High or Low byte)
Destination Register	4x +6	4x register (begin read at High or Low byte)
ASCII String Character Count	4x +7	Defines the search area

Warning! DO NOT modify the address in the middle node of the XMIT block or delete it from the program while it is active. This locks up the port preventing communications.

Bottom Node Content

The bottom node must contain a constant equal to (#0008). This is the number of registers used by the XMIT conversion instruction.

4.2 XMIT Conversion Control Table

The following is a detail description of each of the (8) XMIT conversion control table registers.

4.2.1 XMIT Revision Number (4x) Read Only

Displays the current revision number of XMIT block. This number is automatically loaded by the block and the block over writes any other number entered into this register.

4.2.2 Conversion Fault Status (4x + 1) Read Only

This field displays a fault code generated by the XMIT conversion block. A complete list is shown in the table below.

Fault Status (4x +1)

Fault Code	Fault Description
122	Top node of XMIT not equal to zero, one or two
123	Bottom node of XMIT is not equal to seven, eight or sixteen
131	Invalid character count
135	Invalid destination register block
136	Invalid source register block
137	No ASCII number present
138	Multiple sign characters present
139	Numerical overflow detected
140	String mismatch error
141	String not found
142	Invalid error check detected
143	Invalid conversion opcode

4.2.3 Available to User (4x + 2)

The XMIT conversion block does not use this register. However, it may be used in ladder logic as a pointer. An efficient way to use the XMIT block is to place a pointer value of a TBLK instruction into this register. For a example, refer to Chapter Five, Application Examples.

4.2.4 Data Conversion Control Bits (4x + 3)

This 16 bit word relates to the Data Conversion (4x + 4) word. These bits provide additional control options based on which of the eleven conversions you select.

(4x + 3) Data Conversion Control Bits and their Definitions

Bit #	Definition	1=	0=
1	Not Used		
2	CRC 16 seed	0x0000	0xFFFF
3	Error check type	LRC 8	CRC 16
4	Error check	Validate	Append
5 & 6	Not Used		
7	Conversion case	Upper to lower	Lower to upper
8	Case sensitivity	No	Yes
9	Format leading	Zeros	Blanks
10	Output format	Fixed	Variable
11	Conversion type	Unsigned	Signed
12	Conversion word	32bit	16bit
13	Automatic advance source pointer (points to the next character after the last character purged)	Yes	No
14	Automatic advance destination pointer (points to the next character after the last character purged)	Yes	No
15	Begin reading ASCII at source beginning with ...	Low byte	High byte (normal)
16	Begin saving ASCII at destination beginning with ...	Low byte	high byte (normal)

Certain bits relate to certain conversions. Those bits not mentioned are not validated or modified by the selected conversion and they have no function in relation to the selected conversion. Therefore, they are just ignored.

Conversion to Pertinent Bits Relationship

Conversion Type (opcode)	Pertinent Bit State (software sets bit state)
Illegal opcode (displayed when illegal opcode is detected)	
ASCII decimal to integer (1)	16=0,11,12,13,15 (7=1, 8=0)
ASCII hex to integer (2)	16=0,11,12,13,15 (7=1, 8=0)

ASCII hex to integer array (3)	13,15,16 (none)
Integer to ASCII decimal (4)	15,9,10,11,12,14,16 (none)
Integer to ASCII hex (5)	15,9,10,11,12,14,16 (none)
Integer array to ASCII hex (6)	14,15,16, (8=yes)
Swap source bytes to destination (7)	14,15,16, (8=yes)
Copy source block to destination (8)	7,8,14,15,16 (none)
Compare source & destination blocks (9)	7,8,13,15,16 (none)
Search source block for defined string defined in destination (10)	7,8,13,15,16 (none)
Validate or append error check in source block (11)	2,3,4,13,15 (8=yes, 14=yes, 16=1/0)

4.2.5 Data Conversion Opcodes (4x + 4)

Select the type of conversion you want to perform from the list of eleven options listed in the table. After picking the type of conversion refer to Data Conversion Control Bits (4x + 3) for additional control options that relate to the specific conversion type selected.

(4x + 4) Data Conversion Opcodes and their Definitions

Data Type (4x block)	Action	Data Type (4x block)
Illegal opcode	displayed when illegal opcode is detected	Not applicable
Received ASCII decimal character string (1)	Converted to	16bit or 32bit signed or unsigned binary integer
Received ASCII hex character string (2)		16bit or 32bit unsigned binary integer
Received ASCII hex character string (3)		16bit unsigned binary integer array
16bit or 32bit signed or unsigned integer (4)		ASCII decimal character string for transmission
16bit or 32bit unsigned binary integer (5)		ASCII hex character string for transmission
16bit unsigned integer array (6)		ASCII hex character string for transmission
High and low bytes from saved ASCII source register block (7)	Swapped to	ASCII destination register block
ASCII string from source register block (8)	Copied to	ASCII destination register block with or without case conversion
ASCII source register block (9)	Compared to	ASCII string defined in destination register block with or without case sensitivity
ASCII source register block (10)	Search for	ASCII string defined in destination block with or without case sensitivity
Error check 8bit LRC or 16bit CRC (11)	Validated or Appended on	ASCII string in source register block

☞ **Note:** Binary to BCD and BCD to binary conversions may be performed using more than one XMIT conversion block. For details, refer to Section 4.4, Other Conversion Types.

4.2.6 Source Register (4x + 5)

Enter the 4x register desired. This is the first register in the source block that is read. Ensure you select where you want the read to begin (high or low byte). The selection beside this register in the DX zoom screen is the same as bit 15 in (4x +3).

4.2.7 Destination Register (4x + 6)

Enter the 4x register desired. This is the first register in the destination block that is saved. Ensure you select where you want the save to begin (high or low byte). The selection beside this register in the DX zoom is the same as bit 16 in (4x +3).

4.2.8 ASCII String Character Count (4x + 7)

Enter the search area. This register defines the search area. When either automatic advance source (Bit 13) or automatic advance destination (Bit 14) are ON and no ASCII character is detected, the block automatically adjusts the character count.

4.3 XMIT Conversion Opcode Examples

This section provides you with examples using all eleven conversion options (opcodes 1 ... 11). Individual examples come with complete settings and data information along with a descriptions.

4.3.1 ASCII Input Related Conversion Examples

Opcodes 1 ... 3 convert ASCII input data into binary data. The ASCII input data is received via the PLC port and XMIT communication block. The ASCII data is then converted into binary data. At this point the converted binary data is ready to be used by the PLC based upon your application needs.

These opcodes parse variable length ASCII string data, starting at the source register high or low byte, as selected by (4x +3, Bit 15) data conversion control register. The ASCII string character count register (4x + 7) defines the maximum number of characters that can be parsed from the source string and must initially contain a value between 1 ... 1024. The data conversion control register (4x + 3) also selects the conversion length of 16bit or 32bit (4x + 3, Bit 12) and selects signed or unsigned (4x +3, Bit 13) conversion.

For opcodes 1 ... 3, the initial ASCII string character count (4x + 7) is reduced by the total number of characters parsed from the ASCII source string, and the source string pointer (4x + 3, Bit 13) is advanced to one character past the last character parsed during the conversion.

Note: An error occurs when no hex or decimal digits are present, or when the destination register (4x +6) block is greater than 512 registers or runs past the end of the PLCs state RAM configuration.

ASCII Input Related Conversion Examples

Opcode	Actions	Data (you enter)
1	Source block starting at 400201 high byte =	“–001234567crlf”
	Initial character count=	0x000C
	Conversion control selection=	32bit signed conversion
	32bit signed destination register pair is loaded with=	0xFFED2979
	Source block advanced to 400206 high byte, now aims at=	“crlf”
	ASCII string character count is reduced to=	0x0002
2	Source block starting at 400201 high byte =	“+F301C23 cat”
	Initial character count=	0x000C
	Conversion control selection=	32bit unsigned conversion
	32bit signed destination register pair is loaded with=	0x0FE01C23
	Source block advanced to 400205 high byte, now aims at=	“cat”
	ASCII string character count is reduced to=	0x0004
3	Source block starting at 400301 low byte =	“124ABC0AFcrlf”
	Initial character count=	0x000B
	32bit signed destination register pair is loaded with=	0x0FE01C23
	Source block advanced to 400306 high byte, now aims at=	“crlf”
	ASCII string character count is reduced to=	0x0002

4.3.1.1 Description of Opcode 1 Example

Opcode 1 skips initial white space and then looks for optional sign, “+” or “-”, and at least one decimal digit, “0” to “9”, terminated by something other than white space or decimal digit. Then the binary equivalent value of the string is written into the destination register, for 16bit conversion, or into the destination register pair, for 32bit conversion. The 32bit destination register pair has least significant word (LSW) stored in the lower register number and most significant word (MSW) stored in the higher register number. An error occurs when no decimal digit is present, or when so many digits are present that the converted binary equivalent is too large to fit in the requested storage type.

4.3.1.2 Description of Opcode 2 Example

Opcode 2 skips initial white space and then looks for optional sign, “+” or ”-”, and at least one hex digit, “0” to “9” or “A” to ’F” or “a” to “f”, terminated by something other than white space or hex digit. Then the binary equivalent value of the string is written into the 16bit or 32bit destination.

4.3.1.3 Description of Opcode 3 Example

Opcode 3 converts ASCII hex characters into an array of 16bit binary equivalents, with 4 ASCII characters packed into each 16bit storage word.

4.3.2 ASCII Output Related Conversion Examples

Opcodes 4 ... 6 convert PLC binary data into ASCII data. Once the PLC binary data is converted into ASCII data it is then transmitted via the PLC port and XMIT communication block. At this point the converted ASCII data is ready to be used by the field device based upon your application needs.

☞ **Note:** In opcodes 4 ... 6, an error occurs when the destination register block is greater than 512 registers or runs past the end of the PLCs state RAM configuration.

ASCII Output Related Conversion Examples

Opcode	Actions	Data (you enter)
4	Source contains=	0x9CDE
	Destination block at 400101 high byte	
	Initial character count is=	0x000C
	Conversion control selects 16bit signed fixed output format using leading zeroes	
	Destination block is loaded with=	“000000040158”
	Final character count is=	0x0000
	Destination block advanced to 400107 high byte	
5	Source contains=	0x03FE1234
	Destination block at 400001 low byte	
	Initial character count is=	0x0010
	Conversion control selects 32bit unsigned variable output format	
	Destination block is loaded with=	“3FE1234”

	Final character count is=	0x0009
	Destination block advanced to 400005 high byte	
6	Source contains=	0x5B3D, 0x467E, 0xD14F, 0x478C
	Destination block at 400201 low byte	
	Initial character count is=	0x0007
	Destination block is loaded with=	“5B3D467”
	Final character count is=	0x0000
	Destination block advanced to 400205 high byte	

4.3.2.1 Description of Opcode 4 and 5 Example

Opcodes 4 and 5 generate variable length ASCII output data when the data conversion control register (4x +3, Bit 10) is 0 (variable). Then the number of ASCII output characters generated are subtracted from the initial ASCII string character count register (4x + 7) and the destination pointer (4x + 3, Bit 14) is advanced. When the data conversion control register (4x +3, Bit 10) is 1 (fixed). Then enough leading zeros or blanks, based on the state of the data conversion control register (4x +3, Bit 9) is loaded into the destination register block (4x + 6), in front of the conversion data, to force the total number of characters to be exactly the requested amount. The ASCII string character count (4x + 7) is set to zero and the destination pointer (4x + 3, Bit 14) is advanced. An error occurs when the binary source value generates more decimal characters than can fit in the defined destination register block.

4.3.2.2 Description of Opcode 6 Example

Opcode 6 converts an array of binary registers from the source block into ASCII hex digit characters, that are loaded into the destination block.

4.3.3 ASCII String Related Conversion Examples

Opcodes 7 ... 11 perform five different ASCII string operations within the PLC based upon your application needs. We recommend you define your source and destination blocks using different 4x references that do not overlap.

When using byte swap (opcode 7) or string copy (opcode 8) with case conversion, the source and destination blocks may be the same. When using byte swap (opcode 7) or string copy (opcode 8) the destination block is loaded, the destination pointer (4x + 3, Bit 14) is advanced past

the last character written, and the ASCII string character count (4x + 7) is reduced to zero.

When using string compare (opcode 9) or string search (opcode 10), the source pointer (4x + 3, Bit 13) is advanced, and the ASCII string character count (4x + 7) is reduced.

☞ **Note:** In general, The source pointer auto advance (4x + 3, Bit 13) and the destination pointer auto advance(4x + 3, Bit 14) must be on in the conversion control register (4x + 3), or else these pointers retain their original values, as well as, the initial character count (4x + 7).

ASCII String Related Conversion Examples

Opcode	Actions	Data (you enter)
7	Source contains=	“ABCDEF”
	Destination block at 400001 low byte	
	Initial character count is=	0x0006
	Destination block is loaded with=	“BADCFE”
	Final character count is reduced to=	0x0000
	Destination block advanced to 400004 low byte	
8a	Source contains=	“ABcdeFGH”
	Destination block at 400101 low byte	
	Initial character count is=	0x0006
	Conversion control has case sensitivity on	
	Destination block is loaded with=	“ABcdeF”
	Final character count is reduced to=	0x0000
	Destination block advanced to 400104 low byte	
8b	Source contains=	“abCdeF12”
	Destination block at 400301 high byte	
	Initial character count is=	0x0008
	Conversion control has case sensitivity off, with lower to upper selected	
	Destination block is loaded with=	“ABCDEF12”
	Final character count is reduced to=	0x0000
	Destination block advanced to 400305 high byte	

4.3.3.1 Description of Opcode 7 Example

Opcode 7 uses a source register block of 16bit integers and a destination register block of 16bit integers. Each source word from the source register block is read, bytes swapped and then written into the destination register block. The initial ASCII string character count register (4x + 7) specifies the number of registers to be converted and must be an even number between 2 ... 1024.

4.3.3.2 Description of Opcode 8a and 8b Example

Opcode 8 copies the ASCII string in the source register block into the destination register block. The initial ASCII string character count (4x + 7) specifics the number of characters to be copied. When case sensitivity in the data conversion control register (4x + 3, Bit 8) is off, then the selected lower to upper case or upper to lower case conversion (4x + 3, Bit 7) is performed on the destination block during the copy.

ASCII String Related Conversion Examples

Opcode	Actions	Data (you enter)
9	Destination block contains=	“abcde\0”
	Source block at 400201 high byte contains=	“abcdefgh”
	Initial character count is=	0x0008
	Source block advanced to 400203 low byte, now aims at=	“fgh”
	Final character count is reduced to=	0x0003
	“Operation successful” bottom output goes on because destination string matched in source string	
10	Destination block contains=	“def\0”
	Source block at 400201 high byte contains=	“abcdefgh”
	Initial character count is=	0x0008
	Source block advanced to 400202 low byte, now aims at=	“defgh”
	Final character count is reduced to=	0x0005
	“Operation successful” bottom output goes on because destination string found in source string	
11	Source block at 400201 high byte contains=	0x0103, 0x0001, 0x0008, 0x1234
	Initial character count is=	0x0006
	Conversion control selects LRC8 must be appended	
	Source block at 400201 low byte, now contains=	0x0103, 0x0001, 0x0008, 0xF334

Source block remains at 400201 high byte	
Final character count is increased to=	0x0007
“Operation successful” bottom output goes on because destination string found in source string	

4.3.3.3 Description of Opcode 9 Example

Opcode 9 takes the ASCII string defined in the destination register block and compares it to the source register block. The initial ASCII string character count (4x + 7) specifics the maximum number of characters to be compared, it must be between 1 ... 1024. The match string is contained in the destination block and must be terminated by a 0x00 character. The source pointer (4x + 3, Bit 13) is advanced past the last matching character and the character count (4x + 7) is reduced by the number of characters that matched. When all characters in the source string match the destination string up to the NULL terminator, then the bottom output goes on (operation successful). Otherwise, the middle output goes on (error).

4.3.3.4 Description of Opcode 10 Example

Opcode 10 takes the ASCII string defined in the destination register block and searches the source register block. The initial ASCII string character count (4x + 7) specifics the maximum number of characters to be searched, it must be between 1 ... 1024. The match string is contained in the destination block and must be terminated by a 0x00 character. When the match string is present in the source block, then the source point (4x + 3, Bit 13) is advanced to the start of the matching string. The character count (4x + 7) is reduced by the number of characters skipped over at the beginning of the source block and the bottom output goes on (operation successful). Otherwise, the source pointer and character count are not changed and the middle output goes on (error).

4.3.3.5 Description of Opcode 11 Example

Opcode 11 performs an error check computation for LRC 8bit, CRC 16bit with seed 0xFFFF, or CRC 16bit with seed 0x0000. When conversion control register (4x +3, Bit 4) is set (validate), the selected error check, at the end of the ASCII string in the source block with its given length defined by the ASCII string character count, is validated. When the error check is valid, then the bottom output goes on (operation successful). Otherwise,the middle output goes on (error).

When conversion control register (4x +3, Bit 4) is 0 (append), then the selected error check is computed and appended to the end of the ASCII

string in the source block. The character count is increased by the byte size of the error check, the source pointer is not advanced, and the bottom output goes on (operation successful).

4.4 Other Conversion Types

Additional conversion types are possible using two XMIT conversion blocks.

4.4.1 Binary to BCD Conversion

Two XMIT conversion blocks must be used to perform this conversion type. The first XMIT conversion block uses (opcode 4) to convert the 32bit binary source integer into a 10 digit fixed place ASCII decimal character string saved to a 4x register block. The second XMIT conversion block uses (opcode 2) to convert a hexadecimal ASCII character string read from the same 4x register block, into 32bit BCD destination integer. The binary source integer must be smaller than 0x05F5E0FF, which is 99999999 decimal.

4.4.2 BCD to Binary Conversion

Two XMIT conversion blocks must be used to perform this conversion type. The first XMIT conversion block uses (opcode 5) to convert the 32bit BCD source integer into an 8 digit fixed place ASCII hexadecimal character string saved to a 4x register block. The second XMIT conversion block uses (opcode 1) to convert a decimal ASCII character string read from the same 4x register block, into a 32bit binary destination integer. When all 8 characters are parsed, then the BCD source integer is a valid BDC number.

Chapter 5
Application Examples

- Multiple Modbus Commands Transmission from PLC master to PLC slave
- Fault Word Transmission to Slave PLC via Dialup Modems

5.1 Multiple Modbus Commands Transmission from PLC master to PLC slave

The ladder logic in this example shows how to send multiple Modbus commands to a slave PLC using one XMIT block. The example specifically sends four Modbus commands to a slave PLC with Modbus address #3. The commands perform the following functions:

- Read 25 holding registers (4x) starting at 40010 in slave PLC and place into master PLC starting at 40800.
- Write 25 holding registers (4x) starting at 40825 in master PLC to slave PLC starting at 40010.
- Read 16 coils (0x) starting at 00001 in slave PLC and place into master PLC starting at 00097.
- Write 16 coils (0x) starting at 00113 in master PLC to slave PLC starting at 00001.

5.1.1 Hardware Configuration

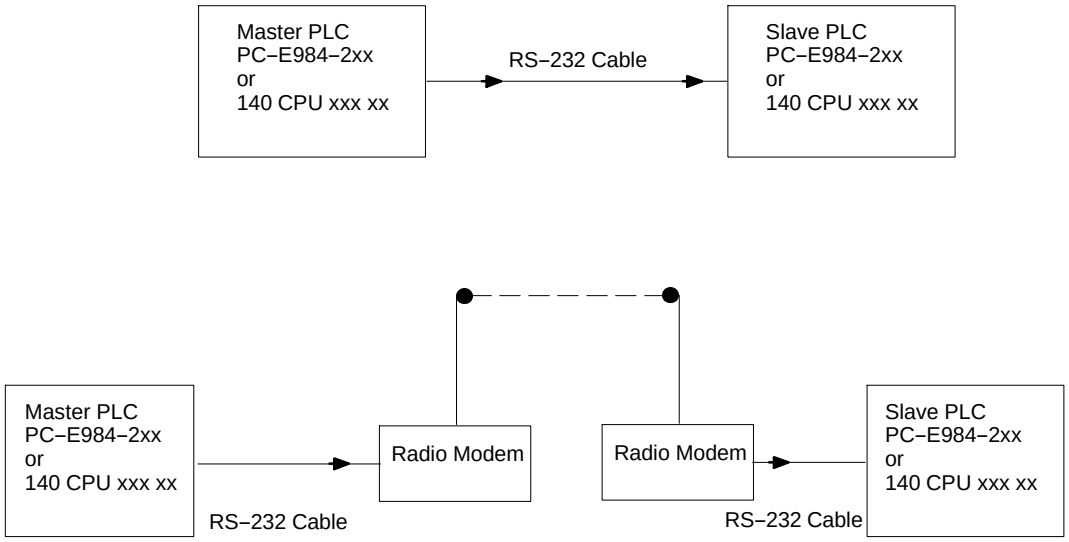


Figure 20 Hardware Configuration for Master to Slave PLC Application

☞ **Note:** This application works with both radio modems and lease line modems.

5.1.2 Master PLC Setup

The XMIT must transmit four Modbus messages from the master PLC port #1. The XMIT forms these messages from the four Modbus message definition tables as defined in the master PLC. The Modbus definition tables are shown below.

Modbus Message Definition Tables

Definition Table #1	Description	Register	Contents
	Modbus Function Code	40100	3
	Quantity	40101	25
	Slave PLC Address	40102	3
	Slave PLC Data Area	40103	10
	Master PLC Data Area	40104	800
Definition Table #2	Description	Register	Contents
	Modbus Function Code	40105	16
	Quantity	40106	25
	Slave PLC Address	40107	3
	Slave PLC Data Area	40108	10
	Master PLC Data Area	40109	825
Definition Table #3	Description	Register	Contents
	Modbus Function Code	40110	1
	Quantity	40111	16
	Slave PLC Address	40112	3
	Slave PLC Data Area	40113	1
	Master PLC Data Area	40114	97
Definition Table #4	Description	Register	Contents
	Modbus Function Code	40115	15
	Quantity	40116	16
	Slave PLC Address	40117	3
	Slave PLC Data Area	40118	1
	Master PLC Data Area	40119	113

5.1.3 Ladder Logic

Network #1 sends the Modbus commands to the slave PLC. The references to holding registers, coils and inputs may be changed based upon your application. Refer to Figure 21.

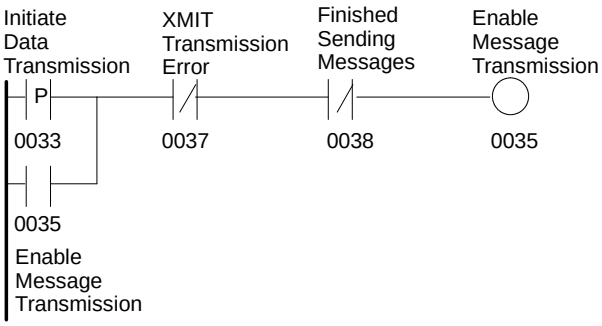


Figure 21 Network #1 Modbus Commands to Slave PLC

Network #1 initiates the Modbus commands to the slave PLC when coil 00033 comes ON. Coil 00035 remains ON until all four Modbus commands are sent to the slave PLC. When an XMIT error occurs during a Modbus transmission to the slave PLC, it unlatches coil 00035.

Network #2 sets up the XMIT control table data (40001 ... 40015) for a new message. Refer to Figure 22.

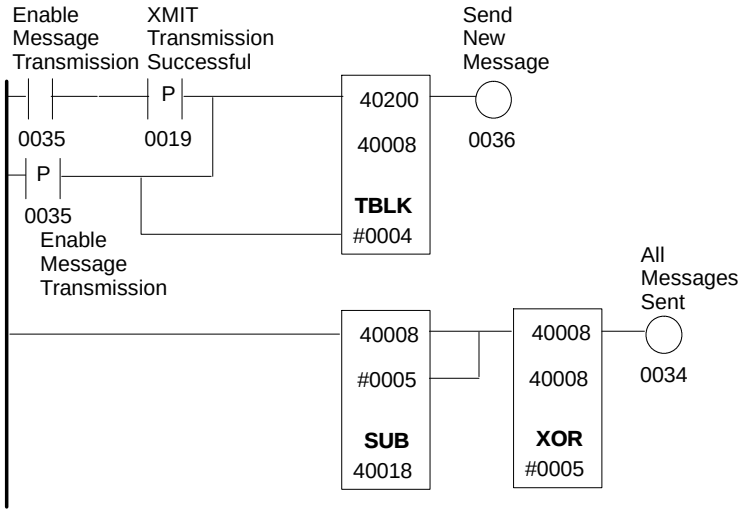


Figure 22 Network #2 Setup XMIT Control Table

Two registers (4x + 2 and 4x + 7) within the XMIT control table (15 register length) are designated as ”Available to User”, so that, pointer values for other instruction blocks like TBLK may be held in these registers. In this example, the TBLK instruction block uses register 40008 (4x + 7) as the pointer.

The TBLK copies data from source tables (see Figure 23 below) into the XMIT control table. In this example, four source tables (Blocks 1 ... Block 4), each four registers long are copied into the XMIT control table, (Destination Block) four registers long (4x + 8 ... 4x + 11). The contents of the source tables (Blocks 1 ... Block 4) and the description of the XMIT control table are shown in the table below.

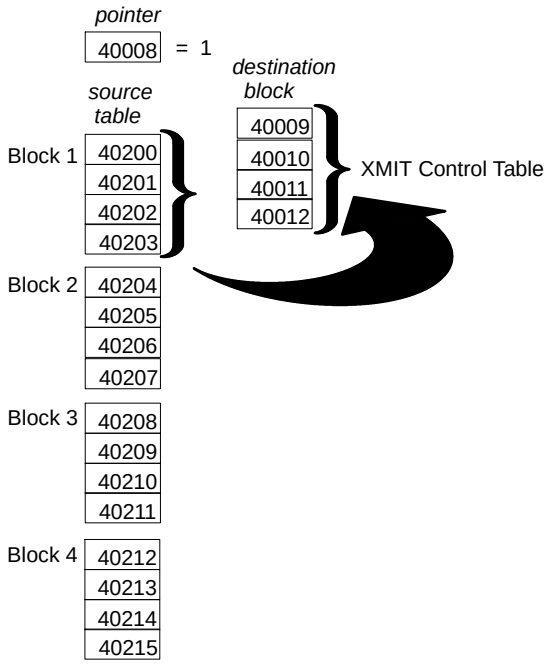


Figure 23 TBLK Operation

Contents of Source Tables and XMIT Control Table

Source Tables	Block 1	40200	00000001 – 00000000 (256 Dec)
		40201	100
		40202	5
		40203	3000
	Block 2	40204	00000001 – 00000000 (256 Dec)
		40205	105
		40206	5
		40207	3000
	Block 3	40208	00000001 – 00000000 (256 Dec)
		40209	110
		40210	5
		40211	3000
	Block 4	40212	00000001 – 00000000 (256 Dec)
		40213	115
		40214	5
		40215	3000
XMIT Control Table	4x + 8	40009	Command Word
	4x + 9	40010	Pointer to Message Table
	4x + 10	40011	Length of Message
	4x + 11	40012	Response timeout (mS)

When coil 00035 goes ON for the first time, TBLK copies the contents of the first source table (Block 1 or 40200 ... 40203) to the XMIT control table (40009 ... 40012). Upon successful completion, the next source table is copied. Thus, TBLK copies the second source table (Block 2 or 40204 ... 40207) to the XMIT control table (40009 ... 40012). The TBLK continues until all four Modbus commands are sent (Block 1 ... Block 4).

The SUB instruction verifies that the source table transfer is complete. It checks after each block transfer.

The XOR instruction clears all registers in the (40009 ... 40012) range.

Network #3 sends the Modbus message from the master PLC to the slave PLC.

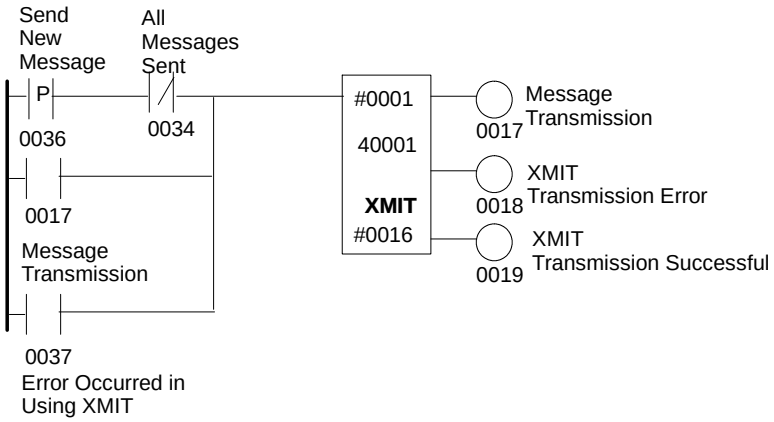


Figure 24 Network #3 Send Modbus Commands Using XMIT

In network #3 the Modbus message is formed using the XMIT instruction so that it may be sent from the master PLC to the slave PLC. The top input of the XMIT instruction remains ON until the Modbus message is successfully sent. The XMIT control table is 16 registers long. In this example, the XMIT control table starts with register 40001 and ends with register 40016. The contents of these registers are shown in the table below.

XMIT Control Table

Description	Register	Value
XMIT Revision Number	40001	201 (or current revision)
Fault Status	40002	0
Available to User	40003	0 (*May be used as pointers for instructions like TBLK)*
Data Rate	40004	9600
Data Bits	40005	8
Parity	40006	0
Stop Bits	40007	1
Available to User	40008	*0 (May be used as pointers for instructions like TBLK)*
Command Word	40009	0000–0001–0000–0000 (256 Dec)
Pointer to Message Table	40010	100

Length of Message	40011	5
Response timeout (mS)	40012	3000
Retry Limit	40013	3
Start of Transmission Delay (mS)	40014	0
End of Transmission Delay (mS)	40015	0
Current retry	40016	0

Network #4 resets the XMIT instruction when a fault occurs.

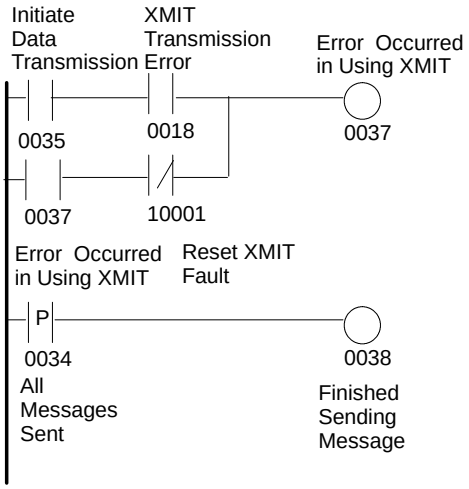


Figure 25 Network #4 Reset XMIT Faults

In network #4 coil 00037 goes ON and remains ON until a reset is performed. As always, based upon your application, you should determine how to address faults and reset your application. Coil 00038 goes ON when all four Modbus commands are successfully sent to the slave PLC. In order to reset (clear the fault) the XMIT instruction block's top input must be turned OFF for one PLC scan.

5.1.4 Conclusion

The four networks of ladder logic in this application example shows how easy it is to send multiple Modbus commands to a slave PLC from a master PLC using only one XMIT instruction. Programming multiple instances of the XMIT control table into the source table of a TBLK, is an excellent method to setup XMIT for a new message. We therefore recommend that you use this method in all future applications implementing the XMIT instruction.

5.2 Fault Word Transmission to Slave PLC via Dialup Modems

The ladder logic in this example shows how to send a single fault word (40800) to a slave PLC using telephone dialup Hayes compatible modems. This logic or a variation of it may be used for applications requiring report by exception fault handling. When a fault occurs, the master PLC uses XMIT to send a dial string to the modem. When the local modem connects to remote modem, the master PLC uses XMIT to send a Modbus message to the slave PLC. The Modbus message writes the contents of fault register (40800) in the master PLC to (40001) in the slave PLC. When the master PLC gets a valid response from the slave PLC, the master PLC uses XMIT to send a hangup string to the local modem. Thus, three messages are transmitted from the master PLC: dial, Modbus command, and hangup.

5.2.1 Hardware Configuration

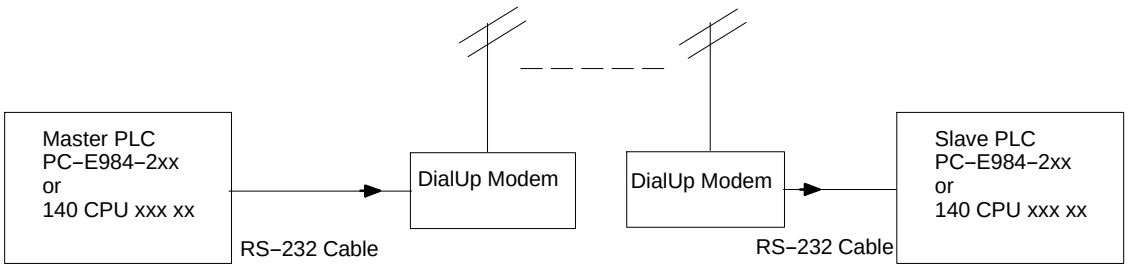


Figure 26 Hardware Configuration for Fault Word Transmission

☞ **Note:** This application works with telephone dialup modems only.

5.2.2 Modem Setup

You must first initialize your dialup modem to ensure proper operation with the XMIT instruction. Program an initialization message or a communication program in the master PLC and send it the the modem via the XMIT function. We recommend using a terminal program to initialize the modem that simplifies the ladder logic. In this example, a communication program named ”Procomm” by DataStrom was used to initialize the modem. When possible, initialize all dialup modem, in the

system, using the same initialization message. The actual initialization message and a definition of each parameter is provided in the table below.

Initialization Message for DialUp Modem

Initialization Message =	AT&F&K0&D0&Q0Q0V1X0
AT=	Attention ***
&F=	Recall factory configuration as active configuration **
&K0=	Disable local flow control **
&D0=	Ignore status of DTR signal **
&Q0=	Communicate in asynchronous mode **
Q0=	Return result codes *
V1=	Display result codes as words *
X4=	Provide basic call progress result codes: *Connect*, *No Carrier*, and *Ring* *
E1	Echo characters from the keyboard to the screen in command state *
<CR>	Carriage return ***
<LF>	Line feed ***
* These parameters *must* always be part of the initialization string for XMIT to function properly.	
** These parameters *should* be part of the initialization string for XMIT to transmit a message to remote modem properly. Only a experienced modem user should change or not use these parameters.	
*** These parameters are automatically added by XMIT, AT before and <CR>, <LF> after, to the message programmed by you.	

☞ **Note:** While some modem manufactures state full compatibility with Hayes, they may still be slightly different. Therefore, we recommend using only those commands that have the same definition as those stated above.

5.2.3 Master PLC Setup

The XMIT must transmit three messages from the master PLC port #1 to the slave PLC: two modem messages (Dial and Hangup), and one modbus message. You must program these messages into the master

PLC holding registers. The actual messages and their content is provided in the table below.

Modem Messages

Dial Message	40150=	68ASC	
	40151=	00ASC	
	40152=	32ASC	
	40153=	6 ASC	
hangup Message	40170=	H0ASC	
Modbus Message	40100=	16	Modbus Function Code
	40101=	1	Quantity
	40102=	3	Slave PLC Address
	40103=	1	Slave PLC Data Area
	40104=	800	Master PLC Data Area

NOTE: The ATDT header and CR/LF trailers are automatically sent and are *NOT* included in the length of message control register (4x+10).

5.2.4 Ladder Logic

Network #1 sends the Modbus commands to the slave PLC. The references to holding registers, coils and inputs may be changed based upon your application. Refer to Figure 27.

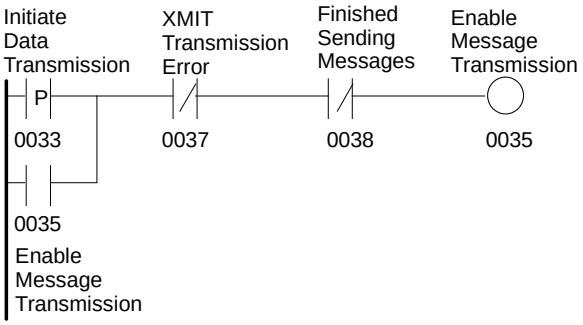


Figure 27 Network #1 Modbus Commands to Slave PLC

Network #1 sends the Modbus commands to the slave PLC when coil 00033 comes ON. Coil 00035 remains ON until all three messages (Modem and Modbus) are sent to the slave PLC. When an XMIT error occurs during a Modbus transmission to the slave PLC, it unlatches coil 00035.

Network #2 sets the XMIT control table (40001 ... 40016) for a new message. Refer to Figure 28.

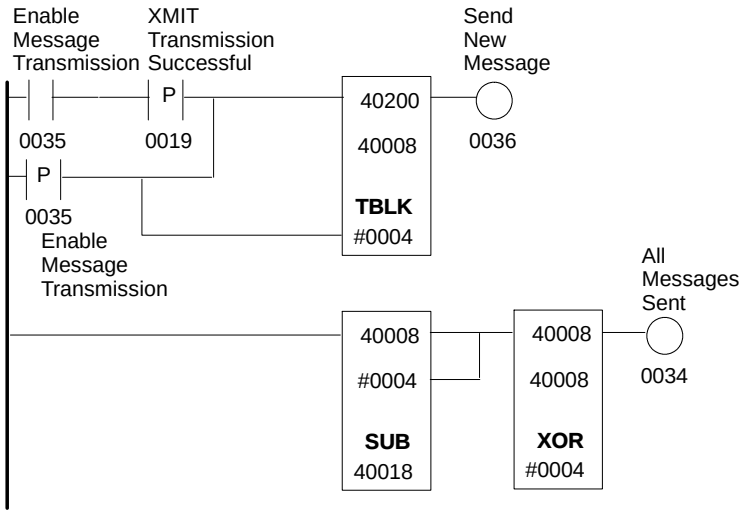


Figure 28 Network #2 Setup XMIT Control Table

Two registers (4x + 2 and 4x + 7) within the XMIT control table (16 register length) are designated as "Available to User", so that, pointer values for other instruction blocks like TBLK may be held in these registers. In this example, the TBLK instruction block uses register 40008 (4x + 7) as the pointer.

The TBLK copies data from source tables (see Figure 29 below) into the XMIT control table. In this example, three source tables (Blocks 1 ... Block 3), each four registers long are copied into the XMIT control table, (Destination Block) four registers long (4x + 8 ... 4x + 11). The contents of the source tables (Blocks 1 ... Block 3) and the description of the XMIT control table are shown in the table below.

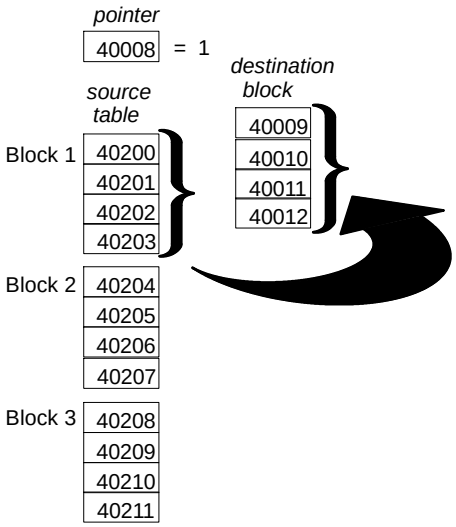


Figure 29 TBLK Operation

Contents of Source Tables and XMIT Control Table

Source Tables	Block 1 *Dial Message:* Sent to modem	40200	00000010 – 00000010 (256 Dec)
		40201	150
		40202	7
		40203	30000
	Block 2 *Modbus Message:* Sent to slave PLC	40204	00000001 – 00000000 (256 Dec)
		40205	100
		40206	5
		40207	3000
	Block 3 *hangup Message:* Sent to Modem	40208	00000010 – 00000100 (256 Dec)
		40209	170
		40210	2
		40211	30000
XMIT Control Table	4x + 8	40009	Command Word
	4x + 9	40010	Pointer to Message Table
	4x + 10	40011	Length of Message
	4x + 11	40012	Response timeout (mS)

Block #1 is the *Dial Message* that is sent to the dialup modem. The first register contains the Command Word. Bit 7 is ON indicating a ASCII message and Bit 15 is ON indicating a dial message. The second register contains a pointer to the dial message starting at (40150). The third register contains the dial message length (7 characters). The fourth register contains the timeout for the dial message (30,000mS). A lot of time is required when a local modem dials a remote modem because a local modem goes through a process to determine a connection. Therefore, we recommend a timeout of approximately 3000mS. When the timeout is too short the XMIT issues a modem reply timeout.

Block #2 is the *Modbus Message* that is sent to the slave PLC. The first register contains the Command Word. Bit 8 is ON indicating a Modbus message. The second register contains a pointer to the Modbus definition table starting at (40100). XMIT uses the information stored here to form a Modbus message. The third register contains the Modbus definition table length (5 registers). The fourth register contains the

timeout for the slave PLC response message (3000mS). The slave PLC response time maybe changed based upon your specific application.

Block #3 is the *hangup Message* that is sent to the slave PLC. The first register contains the Command Word. Bit 14 is ON indicating a hangup message. The second register contains a pointer to the hangup message starting at (40170). The third register contains the hangup message that is two characters long. The fourth register contains the timeout for the hangup message (30,000mS). When the timeout is not long enough, XMIT issues a modem reply timeout. The hangup time maybe changed based upon your specific application.

When coil 00035 goes ON for the first time, TBLK copies the contents of the first source table (Block 1 or 40200 ... 40203) to the XMIT control table (40009 ... 40012). Upon successful completion, the next source table is copied. Thus, TBLK copies the second source table (Block 2 or 40204 ... 40207) to the XMIT control table (40009 ... 40012). The TBLK continues until all three Modbus commands are sent (Block 1 ... Block 3).

The SUB instruction verifies that the source table transfer is complete. It checks after each block transfer.

The XOR instruction clears all registers in the (40009 ... 40012) range.

Network #3 sends the Modbus message from the master PLC to the slave PLC.

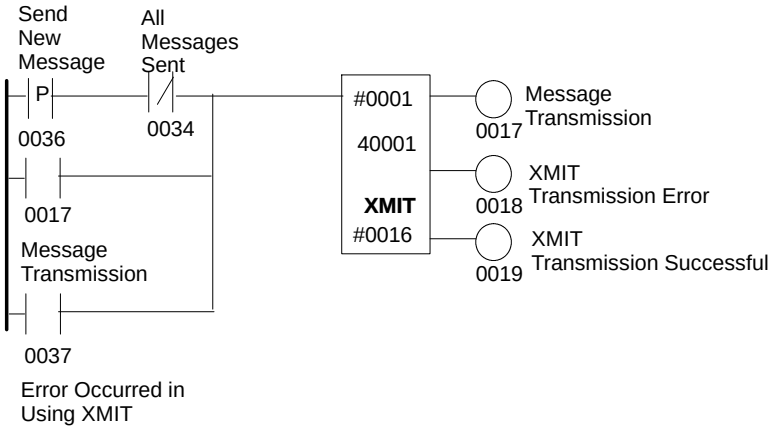


Figure 30 Network #3 Send Modbus Commands Using XMIT

In network #3 the Modbus message is formed using the XMIT instruction so that it may be sent from the master PLC to the slave PLC.

The top input of the XMIT instruction remains ON until the Modbus message is successfully sent. The XMIT control table is 16 registers long. In this example, the XMIT control table starts with register 40001 and ends with register 40016. The contents of these registers are shown in the table below.

XMIT Control Table

Description	Register	Value
XMIT Revision Number	40001	201 (or current revision)
Fault Status	40002	0
Available to User	40003	0 (*May be used as pointers for instructions like TBLK)*
Data Rate	40004	9600
Data Bits	40005	8
Parity	40006	0
Stop Bits	40007	1
Available to User	40008	*0 (May be used as pointers for instructions like TBLK)*
Command Word	40009	0000–0010–0000–0010 (514 Dec)
Pointer to Message Table	40010	150
Length of Message	40011	7
Response timeout (mS)	40012	3000
Retry Limit	40013	3
Start of Transmission Delay (mS)	40014	0
End of Transmission Delay (mS)	40015	0
Current retry	40016	0

Network #4 resets the XMIT instruction when a fault occurs.

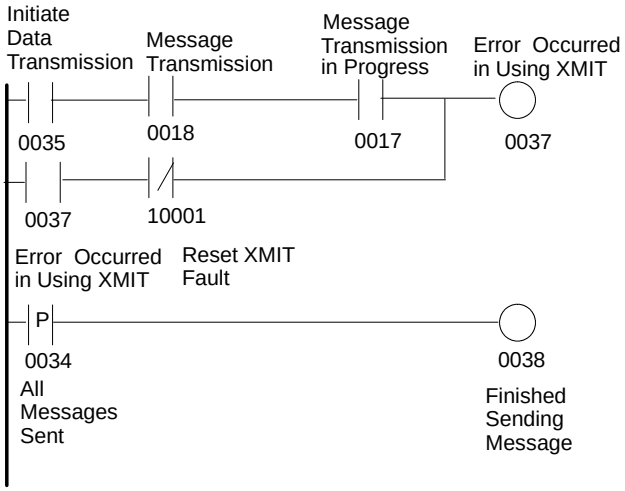


Figure 31 Network #4 Reset XMIT Faults

In network #4 coil 00037 goes ON and remains ON until a reset is performed. As always, based upon your application, you should determine how to address faults and reset your application. Coil 00038 goes ON when all three messages (Modem and Modbus) are successfully sent to the slave PLC. In order to reset (clear the fault) the XMIT instruction blocks top input must be toggled OFF for one PLC scan.

5.2.5 Conclusion

The four networks of ladder logic in this application example shows how easy it is to use a XMIT instruction to communicate between a PLC and a modem. Programmed ASCII messages stored in the master PLC command the modem to dial and hangup. The XMIT sends the message and waits for a reply using the commands you programmed into its control table. Programming multiple instances of the XMIT control table into the source table of a TBLK, is an excellent method to setup XMIT for a new message. We therefore recommend that you use this method in all future applications implementing the XMIT instruction. Also, recall that certain parameters *must* be part of the modem initialization string for XMIT to transmit a message to remote modems properly.

Appendix A
Technical References

- Modbus Query/Response Parameter Limits
- Cable Pinouts
- Cable Adapter Kits
- XMIT Configuration using Hayes Compatible Dialup Modems (Only)

A.1 Modbus Query/Response Parameter Limits

The query/response parameters are limited based upon the type of PLC you are using. Refer to the applicable table below.

884/Quantum PLC Maximum Parameters

Function Code	Description	Query	Response
1	Read Coil Status	2000 Coils	2000 Coils
2	Read Input Status	2000 Inputs	2000 Inputs
3	Read Holding Registers	125 Registers	125 Registers
4	Read Input Registers	125 Registers	125 Registers
5	Force Single Coil	1 Coil	1 Coil
6	Force Single Register	1 Register	1 Register
15	Force Multiple Coil	800 Coils	800 Coils
16	Force Multiple Register	100 Registers	100 Registers
20	Read General References	NOT Supported	NOT Supported
21	Write General References	NOT Supported	NOT Supported

584/984 PLC Maximum Parameters

Function Code	Description	Query	Response
1	Read Coil Status	2000 Coils	2000 Coils
2	Read Input Status	2000 Inputs	2000 Inputs
3	Read Holding Registers	125 Registers	125 Registers
4	Read Input Registers	125 Registers	125 Registers
5	Force Single Coil	1 Coil	1 Coil
6	Force Single Register	1 Register	1 Register
15	Force Multiple Coil	800 Coils	800 Coils
16	Force Multiple Register	100 Registers	100 Registers
20	Read General References (6x)	Maximum length of the entire message can NOT exceed 256 bytes	Maximum length of the entire message can NOT exceed 256 bytes
21	Write General References (6x)	Maximum length of the entire message can NOT exceed 256 bytes	Maximum length of the entire message can NOT exceed 256 bytes

484 PLC Maximum Parameters

Function Code	Description	Query	Response
1	Read Coil Status	512 Coils	512 Coils
2	Read Input Status	512 Inputs	512 Inputs
3	Read Holding Registers	254 Registers	254 Registers
4	Read Input Registers	32 Registers	32 Registers
5	Force Single Coil	1 Coil	1 Coil
6	Force Single Register	1 Register	1 Register
15	Force Multiple Coil	800 Coils	800 Coils
16	Force Multiple Register	60 Registers	60 Registers
20	Read General References	NOT Supported	NOT Supported
21	Write General References	NOT Supported	NOT Supported

184/384 PLC Maximum Parameters

Function Code	Description	Query	Response
1	Read Coil Status	800 Coils	800 Coils
2	Read Input Status	800 Inputs	800 Inputs
3	Read Holding Registers	100 Registers	100 Registers
4	Read Input Registers	100 Registers	100 Registers
5	Force Single Coil	1 Coil	1 Coil
6	Force Single Register	1 Register	1 Register
15	Force Multiple Coil	800 Coils	800 Coils
16	Force Multiple Register	100 Registers	100 Registers
20	Read General References	NOT Supported	NOT Supported
21	Write General References	NOT Supported	NOT Supported

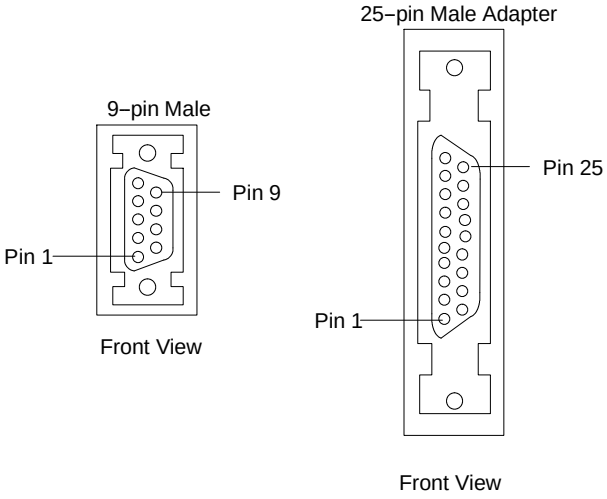
M84 PLC Maximum Parameters

Function Code	Description	Query	Response
1	Read Coil Status	64 Coils	64 Coils
2	Read Input Status	64 Inputs	64 Inputs
3	Read Holding Registers	32 Registers	32 Registers
4	Read Input Registers	4 Registers	4 Registers
5	Force Single Coil	1 Coil	1 Coil
6	Force Single Register	1 Register	1 Register
15	Force Multiple Coil	64 Coils	64 Coils
16	Force Multiple Register	32 Registers	32 Registers

A.2 Cable Pinouts

You need to build an interface cable between your PLC and the modem or printer. The actual cable is connected to Port#1 of the PLC and to the RS232 port of the modem or printer, or direct to another PLC's Modbus port. Because the XMIT supports many modems and printers the pinouts are going to vary. A list of the devices that have been tested with the Modbus master PLC port #1 along with a cable pin--out reference are provided in the *Modicon 309 COM 455 0x XMIT Loadable Read Me First* (GI XMIT RMF). The actual pinouts are provided below.

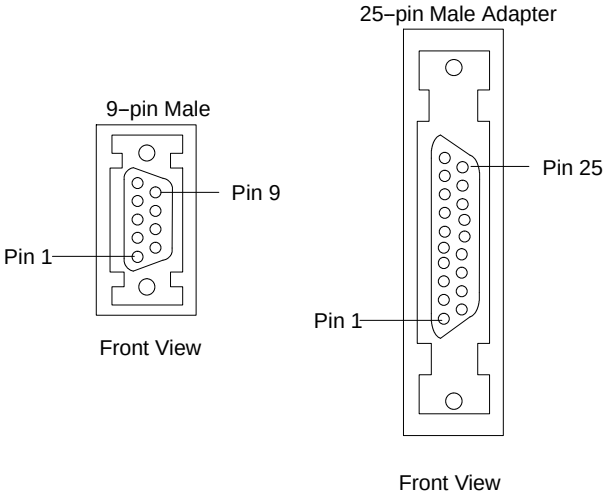
A.2.1 9–pin (RS–232) to 25–pin (Modem) with no RTS/CTS Control



Connector Pinouts

9–Pin Connector			25–Pin D–shell	
RXD	2	⟷	3	RXD
TXD	3	⟷	2	TXD
RTS	7	⟷	4	RTS
CTS	8	⟷	5	CTS
DSR	4	⟷	6	DSR
DTR	6	⟷	20	DTR
GND	5	⟷	7	GND

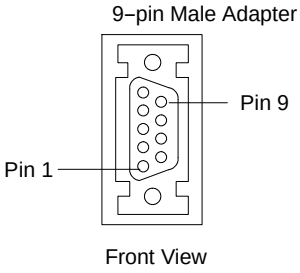
A.2.2 9–pin (RS–232) to 25–pin (Modem) with RTS/CTS Control



Connector Pinouts

9–Pin Connector			25–Pin D–shell	
RXD	2	⟷	3	RXD
TXD	3	⟷	2	TXD
RTS	7	⟷	4	RTS
CTS	8	⟷	5	CTS
DSR	4	⟷	6	DSR
DTR	6	⟷	20	DTR
GND	5	⟷	7	GND

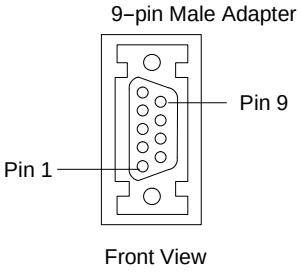
A.2.3 9–pin to 9–pin (Null Modem)



Connector Pinouts

9–Pin Connector			9–Pin Connector	
RXD	2	⟷	3	TXD
TXD	3	⟷	2	RXD
RTS	7	⟷	7	RTS
CTS	8	⟷	8	CTS
DSR	4	⟷	4	DSR
DTR	6	⟷	6	DTR
GND	5	⟷	5	GND

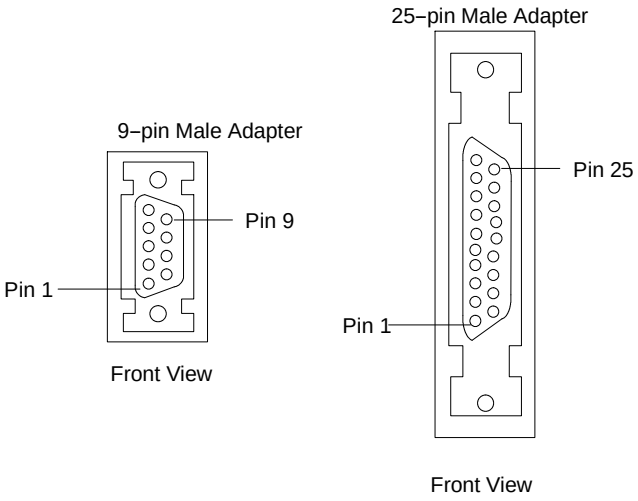
A.2.4 9–pin to 9–pin (Modem)



Connector Pinouts

9–Pin Connector			9–Pin Connector	
TXD	2	◄——►	2	TXD
RXD	3	◄——►	3	RXD
RTS	7	◄——►	7	RTS
CTS	8	◄——►	8	CTS
DSR	4	◄	4	DSR
DTR	6	◄	6	DTR
GND	5	◄——►	5	GND

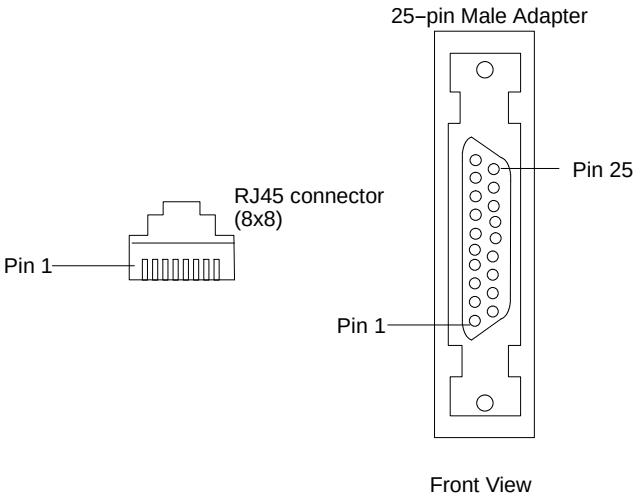
A.2.5 9–pin to 25–pin (Null Modem)



Connector Pinouts

9–Pin Connector			25–Pin D–shell	
RXD	2	◄——►	2	TXD
TXD	3	◄——►	3	RXD
RTS	7	◄	4	RTS
CTS	8	◄	5	CTS
DSR	4	◄	6	DSR
DTR	6	◄	20	DTR
GND	5	◄——►	7	GND

A.2.6 RJ45–(8x8) to 25–pin (Null Modem) 110XCA20401



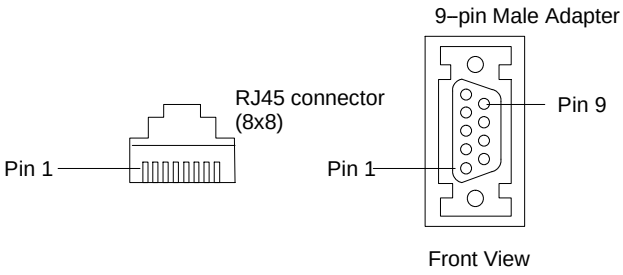
Front View

Connector Pinouts

RJ45 Connector			25–Pin D–shell	
RXD	4	→	2	TXD
TXD	3	→	3	RXD
RTS	6	→	4	RTS
CTS	7	→	5	CTS
GND	5	→	7	GND
DSR	2	→	6	DSR
		→	20	DTR
Chassis Ground	8	→	1	Chassis Ground

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.2.7 RJ45–(8x8) to 9–pin (Null Modem)110XCA20301

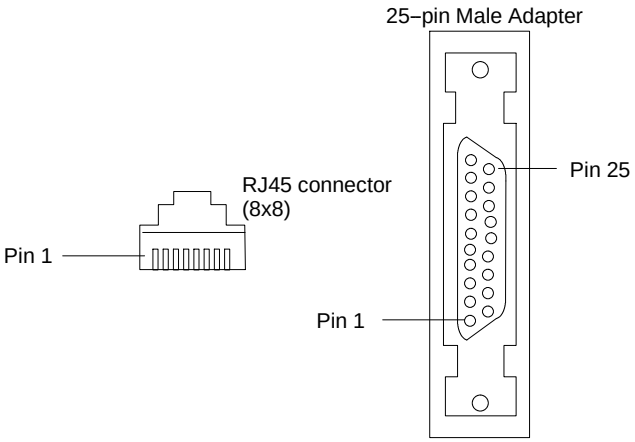


Connector Pinouts

RJ45 Connector			9–Pin D–shell	
RXD	4	◄——►	3	TXD
TXD	3	◄——►	2	RXD
RTS	6	◄	7	RTS
CTS	7	◄	8	CTS
GND	5	◄——►	5	GND
DSR	2	◄——●	4	DTR
		●——►	6	DSR
Chassis Ground	8	◄——►		Case of the Connector

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.2.8 RJ45–(8x8) to 25–pin (Modem)110XCA20401

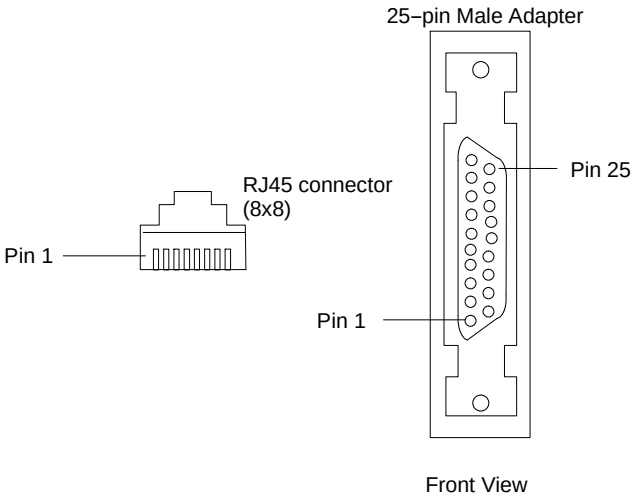


Connector Pinouts

RJ45 Connector			25–Pin D–shell	
RXD	4	→	3	RXD
TXD	3	→	2	TXD
RTS	6	→	4	RTS
CTS	7	→	5	CTS
GND	5	→	7	GND
DSR	2	→	6	DSR
		→	20	DTR
Chassis Ground	8	→	1	Chassis Ground

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.2.9 RJ45–(8x8) to 25–pin (Modem)110XCA20401

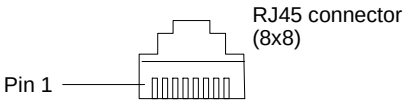


Connector Pinouts

RJ45 Connector			25–Pin D–shell	
RXD	4	◄——►	3	RXD
TXD	3	◄——►	2	TXD
RTS	6	◄——►	4	RTS
CTS	7	◄——►	5	CTS
GND	5	◄——►	7	GND
			6	DSR
		►——►	20	DTR
Chassis Ground	8	◄——►	1	Chassis Ground

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.2.10 RJ45–(8x8) to RJ45–(8x8) (Modem)

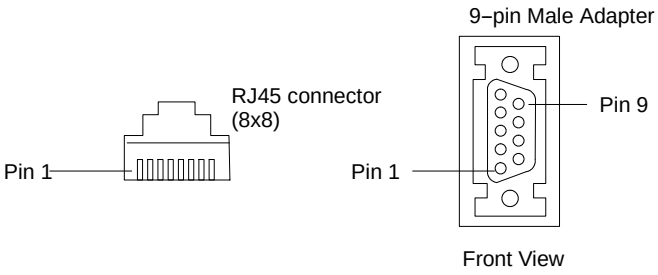


Connector Pinouts

RJ45 Connector			RJ45 Connector	
RXD	4	◄——►	4	RXD
TXD	3	◄——►	3	TXD
RTS	6	◄——►	6	RTS
CTS	7	◄——►	7	CTS
GND	5	◄——►	5	GND
DSR	2	◄——►	2	DSR
Chassis Ground	8	◄——►	8	Chassis Ground

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.2.11 9–pin to RJ45–(8x8) (Modem)110XCA20301

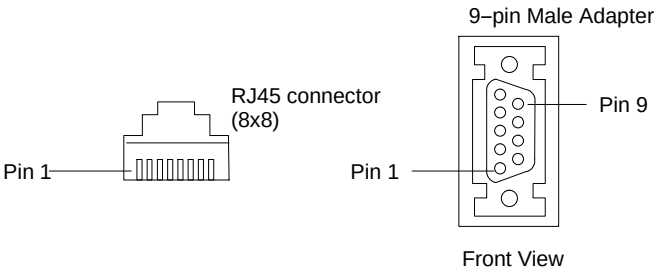


Connector Pinouts

RJ45 Connector			9–Pin Connector	
RXD	4	◄——►	2	RXD
TXD	3	◄——►	3	TXD
RTS	6	◄——►	7	RTS
CTS	7	◄——►	8	CTS
GND	5	◄——►	5	GND
DSR	2	◄——►	6	DSR
		◄——►	4	DTR
Chassis Ground	8	◄——►		Case of the Connector

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.2.12 9–pin to RJ45–(8x8) (Modem)110XCA20301



Connector Pinouts

RJ45 Connector			9–Pin Connector	
RXD	4	◄——►	2	RXD
TXD	3	◄——►	3	TXD
RTS	6	◄——►	7	RTS
CTS	7	◄——►	8	CTS
GND	5	◄——►	5	GND
			6	DSR
		——►	4	DTR
Chassis Ground	8	◄——►		Case of the Connector

Caution: Pin1 of the RJ45 receives 5V from the PLC.

A.3 Cable Adapter Kits

You may want to purchase Cable Adapter Kits for your RJ45 (8x8) requirements rather than make them. The table below provides a list of available kits.

Cable Adapter Kits

Description	Modicon Part Number
RJ45–(8x8) to 25–Pin (Male)	110XCA20401
RJ45–(8x8) to 9–Pin (Male)	110XCA20301
RJ45–(8x8) to 9–Pin (Female)	110XCA20302
RJ45–(8x8) to 25–Pin (Female)	110XCA20402

A.4 XMIT Configuration using Hayes Compatible Dial–Up Modems (Only)

There are three commands that you need to become familiar with when interfacing dial-up modems to XMIT. These commands are: Initialize modem, Dial modem, and Hangup modem. Before an ASCII message or a Modbus message goes through the modem, you must first send an initialization string and then a dial string to the modem. Once the modem has dialed the telephone number and made a connection to the remote modem, you may send an unlimited number of ASCII messages or Modbus messages through the modem. To send multiple messages, you increment the message pointer to the next message after each successful XMIT operation. When all messages are sent, you may then send the hangup string to the modem.

A.4.1 Initialization Message

The initialization message is just like any other ASCII message and may be a maximum of 512 characters long, although 50 characters is usually more than enough to initialize a modem. You may implement any Hayes AT command as part of the initialization string. We recommend the following commands when initializing a modem for use with XMIT.

Initialization Message for Dial–Up Modem

Initialization Message =	AT&F&K0&Q0&D0V1Q0X0E1
AT=	Self–calibrate Modem *
&F=	Recall factory configuration as active configuration *
&K0=	Disable local flow control **
&Q0=	Communicate in asynchronous mode **
&D0=	Ignore status of DTR signal *
V1=	Display result codes as words * If V1 is not used or if modem is not capable of returning verbose responses the XMIT block returns error 117 (modem replay time out).
Q0=	Return result codes *
X4=	Provide basic call progress result codes: *Connect*, *No Carrier*, and *Ring* *

E1=	Echo characters from the keyboard to the screen in command state *

* These parameters *must* always be part of the initialization string for XMIT to function properly.

** These parameters *should* be part of the initialization string for XMIT to transmit a message to remote modem properly. Only a experienced modem user should change or not use these parameters.

☞ **Note:** While some modem manufactures state full compatibility with Hayes, they may still be slightly different. Therefore, we recommend using only those commands that have the same definition as those stated above.

The initialization message must always start with Hayes standard AT command. The XMIT block automatically precedes modem command messages with AT and appends the message with carriage return (0x0D) and line feed (0x0A) characters since these are required by all modem control messages. Other (non controlling) ASCII messages do not have to end with a carriage return and line feed.

For example, a typical initialization message that XMIT sends to the modem.

Message = (AT)&F&K0&Q0&D0V1X0Q0 (<CR><LF>) *
Length = 17 characters

For example, the initialization message may also be used to set S--registers of the modem.

Message = (AT)S0=1 (<CR><LF>) *
Length = 4 characters

*Characters within parentheses are automatically sent.

To have XMIT send an initialization message to the modem, bit 7 and bit 16 of the command word must be ON. When bit 16 is ON, bits 15 and 14 must not be ON or XMIT will not complete the operation successfully. To actually send the message, the top input of XMIT must come ON and stays ON until the operation is complete or an error occurs. When XMIT determines the message was successfully sent to the modem, it turns ON the bottom output. When an error occurs, the middle output

comes ON. The top output is ON while the message is being sent to the modem.

Expert: To eliminate some ladder logic programming, you may initialize the modem with parameters via a terminal program and not use XMIT. Once the parameters are in the modem memory they may be saved to non-memory with an AT command, usually &W.

A.4.2 Dial Message

The dial message is used to send a telephone number to the modem. Only AT commands related to dialing a number should be included with the message. Examples of typical dial messages used with XMIT are shown below.

For example, dial telephone number using tone dialing.

Message = (AT)DT)6800326 (<CR><LF>) *
Length = 7 characters

For example, dial telephone number using pulse dialing.

Message = (AT)DP)6800326 (<CR><LF>) *
Length = 7 characters

For example, dial telephone number using tone dialing, wait to hear dial tone before dialing number, and pause before dialing the rest of the number.

Message = (AT)DT)W,6800326 (<CR><LF>) *
Length = 9 characters

*Characters within parentheses are automatically sent.

To have XMIT send a tone dial message to the modem, bit 7 and bit 15 of the command word must be ON. When bit 15 is ON, bits 16 and 14 must not be ON or XMIT will not complete the operation successfully. To actually send the message, the top input of XMIT must come ON and stays ON until the operation is complete or an error occurs. When XMIT determines the message was successfully sent to the modem, it turns ON the bottom output. When an error occurs, the middle output comes ON. The top output is ON while the message is being sent to the modem.

Expert: Because it takes so long for a local modem to make a connection to a remote modem, the timeout value, in register (4x + 11) should be as very long when sending a dial message to a modem. For example, set the timeout for 30,000 mS when sending a dial message. When the timeout value is too short, XMIT issues a message timeout. You may have to try several settings before finding the optimal time.

A.4.3 Hangup Message

The hangup message is used to hangup the modem. Only AT commands related to hanging up the modem should be used in this message. An example of a typical hangup message is shown below.

For example, hangup modem message.

Message = (+ + +AT)H0 (<CR> <LF>) *
Length = 2 characters

*Characters within parentheses are automatically sent.

When the hangup message is sent to a modem that is already connected to a remote modem, XMIT must first set the local modem in command mode. XMIT does this by sending a escape sequence + + + to the modem. XMIT assumes that + + + sets the modem in command mode. Some modem manufactures let the owner change this default escape sequence. For XMIT to function properly the modem should be set to accept the + + + escape sequence.

To have XMIT send a hangup message to the modem, bit 7 and bit 14 of the command word must be ON. When bit 14 is ON, bits 16 and 15 must not be ON or XMIT will not complete the operation successfully. To actually send the message, the top input of XMIT must come ON and stays ON until the operation is complete or an error occurs. When XMIT determines the message was successfully sent to the modem, it turns ON the bottom output. When an error occurs, the middle output comes ON. The top output is ON while the message is being sent to the modem.

Expert: Because it takes so long for a local modem to hangup once it receives the hangup command, the timeout value, in register (4x + 11) should be as very long when sending a dial message to a modem. For example, set the timeout for 30,000 mS when sending a dial message. When the timeout value is too short, XMIT issues a message timeout. You may have to try several settings before finding the optimal time.

Index

Numbers

1 through 6, 15 and 16, Modbus function codes, 40
110XCA20301, Cable Adapter Kit, 99, 103, 104
110XCA20401, Cable Adapter Kit, 98, 100, 101
184/384, PLC parameter limits, 91
20 and 21, Modbus function codes, 43
484, PLC parameter limits, 91
584/984, PLC parameter limits, 90
8, Modbus function codes, 42
884/Quantum, PLC parameter limits, 90
9-pin (RS-232) to 25-pin (Modem), Modem cable pin-outs, 93, 94
9-pin to 25-pin (Null Modem), Modem cable pin-outs, 97
9-pin to 9-pin (Modem), Modem cable pin-outs, 96
9-pin to 9-pin (Null Modem), Modem cable pin-outs, 95
9- pin to RJ45-(8x8) (Modem)
  110XCA20301, Modem cable pin-outs, 103, 104

A

Application example
  Radio/lease line modem, 72
  Telephone dial-up modem, 80

B

Bit definitions, Command word, 31

C

Cable Adapter Kit
  110XCA20301, 99, 103, 104
  110XCA20401, 98, 100, 101
Cable Adapter Kits-Available, Quick Reference, 105
Command word, Bit definitions, 31
Communication control table, Valid ranges, 25
Conversion control table, Valid ranges, 55

D

Data rate ranges, Data rate register, 29
Data rate register, Data rate ranges, 29
DXFDT.SYS, Installation, 7

F

Fault codes, Fault status register, 50
Fault codes (XMIT communication), Fault status register (XMIT communication), 27
Fault codes (XMIT conversion), Fault status register (XMIT conversion), 57
Fault status register (XMIT communication), Fault codes (XMIT communication), 27
Fault status register (XMIT conversion), Fault codes (XMIT conversion), 57
Fault status register (XMIT port status), Fault codes (XMIT port status), 50

I

Installation
  DXFDT.SYS file, 7
  NSUP.EXE file, 8
  XMIT.EXE file, 9
  XMIT.ZMM, 7
  XMIT1968.HLP, 8

M

M84, PLC parameter limits, 92
Modbus function codes
  1 through 6, 15 and 16, 40
  20 and 21, 43
  8, 42
Modem cable pin-outs
  9-pin (RS-232) to 25-pin (Modem), 93, 94
  9-pin to 25-pin (Null Modem), 97
  9-pin to 9-pin (Modem), 96
  9-pin to 9-pin (Null Modem), 95
  9-pin to RJ45-(8x8) (Modem)
    110XCA20301, 103, 104

RJ45-(8x8) to 25-pin (Modem)
110XCA20401, 100, 101
RJ45- (8x8) to 25-pin (Null Modem)
110CXA20401, 98
RJ45-(8x8) to 9-pin (Null Modem)
110XCA20301, 99
RJ45-(8x8) to RJ45-(8x8) (Modem), 102
Modsoft zoom screens
XMIT communication
Control table, 14, 15
Fault codes, 15, 16, 17
Modbus function codes, 13, 14
XMIT conversion, 19
Fault codes, 20
XMIT port status, 18
Fault codes, 18, 19

N

NSUP.EXE, Installation, 8

P

PLC parameter limits
184/384, 91
484, 91
584/984, 90
884/Quantum, 90
M84, 92
Port status control table, Valid ranges, 49

Q

Quick Reference, Cable Adapter Kits-Available, 105

R

Radio/lease line modem, Application example, 72
RJ45-(8x8) to 25-pin (Modem)
110XCA20401, Modem cable pin-outs, 100, 101
RJ45-(8x8) to 25-pin (Null Modem)
110XCA20401, Modem cable pin-outs, 98
RJ45-(8x8) to 9-pin (Null Modem)
110XCA20301, Modem cable pin-outs, 99
RJ45-(8x8) to RJ45-(8x8) (Modem), Modem cable pin-outs, 102

T

Telephone dial-up modem, Application example, 80

V

Valid ranges
Communication control table, 25
Conversion control table, 55
Port status control table, 49

X

XMIT communication, Modsoft zoom screens
Control table, 14, 15
Fault codes, 15, 16, 17
Modbus function codes, 13, 14
XMIT conversion, Modsoft zoom screens, 19
Fault Codes, 20
Fault codes, 20
XMIT port status, Modsoft zoom screens, 18
Fault codes, 18, 19
XMIT.EXE, Installation, 9
XMIT.ZMM, Installation, 7
XMIT1968.HLP, Installation, 8